

[Start Here](#)[Blog](#)[Books](#)[About](#)[Contact](#)

Need help with LSTMs in Python? [Take the FREE Mini-Course.](#)

Multi-step Time Series Forecasting with Long Short-Term Memory Networks in Python

by **Jason Brownlee** on May 10, 2017 in **Long Short-Term Memory Networks**



The Long Short-Term Memory network or LSTM is a recurrent neural network that can learn and forecast long sequences.

A benefit of LSTMs in addition to learning long sequences is that they can learn to make a one-shot multi-step forecast which may be useful for time series forecasting.

A difficulty with LSTMs is that they can be tricky to configure and it can require a lot of preparation to get the data in the right format for learning.

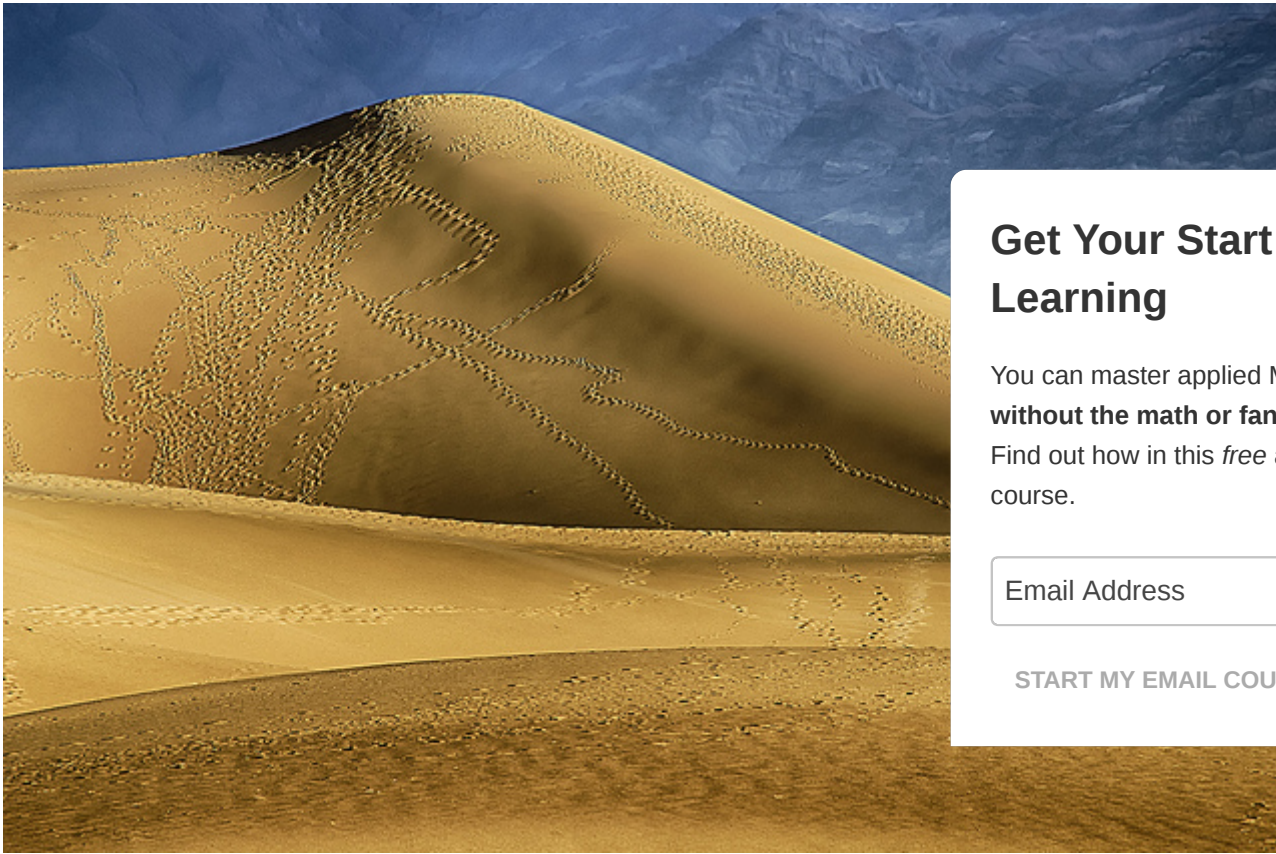
In this tutorial, you will discover how you can develop an LSTM for multi-step time series forecasting

[Get Your Start in Machine Learning](#)

After completing this tutorial, you will know:

- How to prepare data for multi-step time series forecasting.
- How to develop an LSTM model for multi-step time series forecasting.
- How to evaluate a multi-step time series forecast.

Let's get started.



Multi-step Time Series Forecasting with Long Short-Term Memory Networks in Python
Photo by [Tom Babich](#), some rights reserved.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Tutorial Overview

Get Your Start in Machine Learning

This tutorial is broken down into 4 parts; they are:

1. Shampoo Sales Dataset
2. Data Preparation and Model Evaluation
3. Persistence Model
4. Multi-Step LSTM

Environment

This tutorial assumes you have a Python SciPy environment installed. You can use either Python 2 or 3 with this example.

This tutorial assumes you have Keras v2.0 or higher installed with either the TensorFlow or Theano backend.

This tutorial also assumes you have scikit-learn, Pandas, NumPy, and Matplotlib installed.

If you need help setting up your Python environment, see this post:

- [How to Setup a Python Environment for Machine Learning and Deep Learning with Anaconda](#)

Next, let's take a look at a standard time series forecasting problem that we can use as context for the

Need help with LSTMs for Sequence Prediction?

Take my free 7-day email course and discover 6 different LSTM architectures.

Click to sign-up and also get a free PDF Ebook version of the course.

Start Your FREE Mini-Course Now!

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Shampoo Sales Dataset

Get Your Start in Machine Learning

This dataset describes the monthly number of sales of shampoo over a 3-year period.

The units are a sales count and there are 36 observations. The original dataset is credited to Makridakis, Wheelwright, and Hyndman (1998).

You can download and learn more about the dataset [here](#).

The example below loads and creates a plot of the loaded dataset.

```
1 # load and plot dataset
2 from pandas import read_csv
3 from pandas import datetime
4 from matplotlib import pyplot
5 # load dataset
6 def parser(x):
7     return datetime.strptime('190'+x, '%Y-%m')
8 series = read_csv('shampoo-sales.csv', header=0, parse_dates=[0], index_col=0, squeeze=True)
9 # summarize first few rows
10 print(series.head())
11 # line plot
12 series.plot()
13 pyplot.show()
```

Running the example loads the dataset as a Pandas Series and prints the first 5 rows.

```
1 Month
2 1901-01-01    266.0
3 1901-02-01    145.9
4 1901-03-01    183.1
5 1901-04-01    119.3
6 1901-05-01    180.3
7 Name: Sales, dtype: float64
```

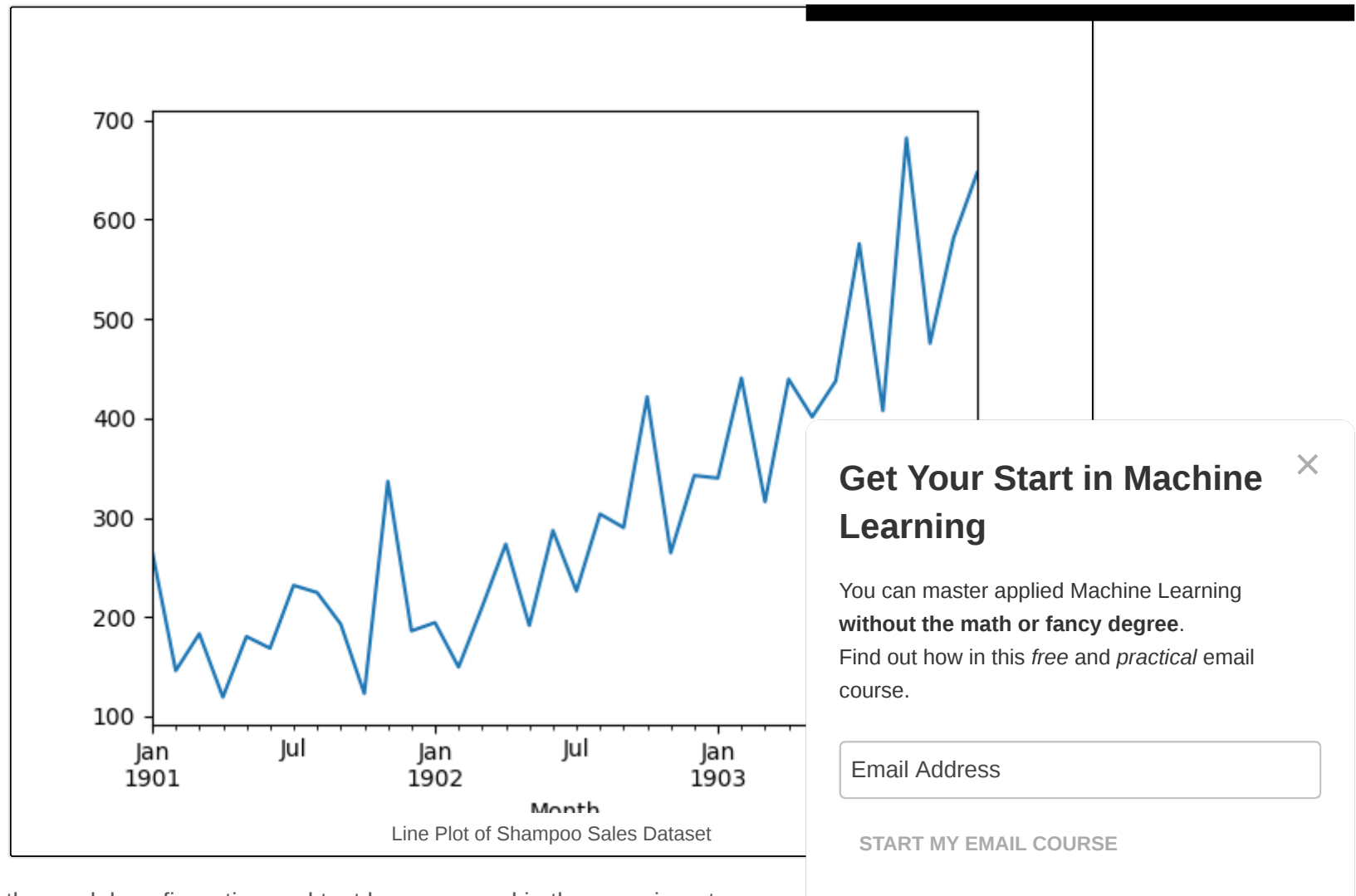
A line plot of the series is then created showing a clear increasing trend.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning



Next, we will take a look at the model configuration and test harness used in the experiment.

Data Preparation and Model Evaluation

This section describes data preparation and model evaluation used in this tutorial

Data Split

Get Your Start in Machine Learning

We will split the Shampoo Sales dataset into two parts: a training and a test set.

The first two years of data will be taken for the training dataset and the remaining one year of data will be used for the test set.

Models will be developed using the training dataset and will make predictions on the test dataset.

For reference, the last 12 months of observations are as follows:

```
1 "3-01", 339.7
2 "3-02", 440.4
3 "3-03", 315.9
4 "3-04", 439.3
5 "3-05", 401.3
6 "3-06", 437.4
7 "3-07", 575.5
8 "3-08", 407.6
9 "3-09", 682.0
10 "3-10", 475.3
11 "3-11", 581.3
12 "3-12", 646.9
```

Multi-Step Forecast

We will contrive a multi-step forecast.

For a given month in the final 12 months of the dataset, we will be required to make a 3-month forecast.

That is given historical observations ($t-1$, $t-2$, ... $t-n$) forecast t , $t+1$ and $t+2$.

Specifically, from December in year 2, we must forecast January, February and March. From January to an October, November, December forecast from September in year 3.

A total of 10 3-month forecasts are required, as follows:

```
1 Dec, Jan, Feb, Mar
2 Jan, Feb, Mar, Apr
3 Feb, Mar, Apr, May
4 Mar, Apr, May, Jun
5 Apr, May, Jun, Jul
6 May, Jun, Jul, Aug
```

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

7	Jun,	Jul,	Aug,	Sep
8	Jul,	Aug,	Sep,	Oct
9	Aug,	Sep,	Oct,	Nov
10	Sep,	Oct,	Nov,	Dec

Model Evaluation

A rolling-forecast scenario will be used, also called walk-forward model validation.

Each time step of the test dataset will be walked one at a time. A model will be used to make a forecast for the time step, then the actual expected value for the next month from the test set will be taken and made available to the model for the forecast on the next time step.

This mimics a real-world scenario where new Shampoo Sales observations would be available each month and used in the forecasting of the following month.

This will be simulated by the structure of the train and test datasets.

All forecasts on the test dataset will be collected and an error score calculated to summarize the skill. The root mean squared error (RMSE) will be used as it punishes large errors and results in a score that is easy to interpret in the context of monthly shampoo sales.

Persistence Model

A good baseline for time series forecasting is the persistence model.

This is a forecasting model where the last observation is persisted forward. Because of its simplicity,

You can learn more about the persistence model for time series forecasting in the post:

- [How to Make Baseline Predictions for Time Series Forecasting with Python](#)

Prepare Data

The first step is to transform the data from a series into a supervised learning problem.

That is to go from a list of numbers to a list of input and output patterns. We can achieve this using a

Get Your Start in Machine Learning

d()).

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

For more on this function, see the post:

- [How to Convert a Time Series to a Supervised Learning Problem in Python](#)

The function is listed below.

```

1 # convert time series into supervised learning problem
2 def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
3     n_vars = 1 if type(data) is list else data.shape[1]
4     df = DataFrame(data)
5     cols, names = list(), list()
6     # input sequence (t-n, ... t-1)
7     for i in range(n_in, 0, -1):
8         cols.append(df.shift(i))
9         names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
10    # forecast sequence (t, t+1, ... t+n)
11    for i in range(0, n_out):
12        cols.append(df.shift(-i))
13        if i == 0:
14            names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
15        else:
16            names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
17    # put it all together
18    agg = concat(cols, axis=1)
19    agg.columns = names
20    # drop rows with NaN values
21    if dropnan:
22        agg.dropna(inplace=True)
23    return agg

```

The function can be called by passing in the loaded series values an *n_in* value of 1 and an *n_out* value of 3:

```

1 supervised = series_to_supervised(raw_values, 1, 3)

```

Next, we can split the supervised learning dataset into training and test sets.

We know that in this form, the last 10 rows contain data for the final year. These rows comprise the test set and the rest of the data makes up the training dataset.

We can put all of this together in a new function that takes the loaded series and some parameters and returns a train and test set ready for modeling.

```

1 # transform series into train and test sets for supervised learning

```

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE


```

2 def prepare_data(series, n_test, n_lag, n_seq):
3     # extract raw values
4     raw_values = series.values
5     raw_values = raw_values.reshape(len(raw_values), 1)
6     # transform into supervised learning problem X, y
7     supervised = series_to_supervised(raw_values, n_lag, n_seq)
8     supervised_values = supervised.values
9     # split into train and test sets
10    train, test = supervised_values[0:-n_test], supervised_values[-n_test:]
11    return train, test

```

We can test this with the Shampoo dataset. The complete example is listed below.

```

1 from pandas import DataFrame
2 from pandas import concat
3 from pandas import read_csv
4 from pandas import datetime
5
6 # date-time parsing function for loading the dataset
7 def parser(x):
8     return datetime.strptime('190'+x, '%Y-%m')
9
10 # convert time series into supervised learning problem
11 def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
12     n_vars = 1 if type(data) is list else data.shape[1]
13     df = DataFrame(data)
14     cols, names = list(), list()
15     # input sequence (t-n, ... t-1)
16     for i in range(n_in, 0, -1):
17         cols.append(df.shift(i))
18         names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
19     # forecast sequence (t, t+1, ... t+n)
20     for i in range(0, n_out):
21         cols.append(df.shift(-i))
22         if i == 0:
23             names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
24         else:
25             names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
26     # put it all together
27     agg = concat(cols, axis=1)
28     agg.columns = names
29     # drop rows with NaN values
30     if dropnan:
31         agg.dropna(inplace=True)
32     return agg
33
34 # transform series into train and test sets for supervised learning

```

Get Your Start in Machine Learning ×

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

```

35 def prepare_data(series, n_test, n_lag, n_seq):
36     # extract raw values
37     raw_values = series.values
38     raw_values = raw_values.reshape(len(raw_values), 1)
39     # transform into supervised learning problem X, y
40     supervised = series_to_supervised(raw_values, n_lag, n_seq)
41     supervised_values = supervised.values
42     # split into train and test sets
43     train, test = supervised_values[0:-n_test], supervised_values[-n_test:]
44     return train, test
45
46 # load dataset
47 series = read_csv('shampoo-sales.csv', header=0, parse_dates=[0], index_col=0, squeeze=True, date_parser=parser)
48 # configure
49 n_lag = 1
50 n_seq = 3
51 n_test = 10
52 # prepare data
53 train, test = prepare_data(series, n_test, n_lag, n_seq)
54 print(test)
55 print('Train: %s, Test: %s' % (train.shape, test.shape))

```

Running the example first prints the entire test dataset, which is the last 10 rows. The shape and size

```

1 [[ 342.3  339.7  440.4  315.9]
2  [ 339.7  440.4  315.9  439.3]
3  [ 440.4  315.9  439.3  401.3]
4  [ 315.9  439.3  401.3  437.4]
5  [ 439.3  401.3  437.4  575.5]
6  [ 401.3  437.4  575.5  407.6]
7  [ 437.4  575.5  407.6  682. ]
8  [ 575.5  407.6  682.   475.3]
9  [ 407.6  682.   475.3  581.3]
10 [ 682.   475.3  581.3  646.9]]
11 Train: (23, 4), Test: (10, 4)

```

We can see the single input value (first column) on the first row of the test dataset matches the observation in the shampoo-sales for December in the 2nd year:

```
1 "2-12",342.3
```

We can also see that each row contains 4 columns for the 1 input and 3 output values in each observation.

Make Forecasts

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

The next step is to make persistence forecasts.

We can implement the persistence forecast easily in a function named *persistence()* that takes the last observation and the number of forecast steps to persist. This function returns an array containing the forecast.

```
1 # make a persistence forecast
2 def persistence(last_ob, n_seq):
3     return [last_ob for i in range(n_seq)]
```

We can then call this function for each time step in the test dataset from December in year 2 to September in year 3.

Below is a function *make_forecasts()* that does this and takes the train, test, and configuration for the dataset as arguments and returns a list of forecasts.

```
1 # evaluate the persistence model
2 def make_forecasts(train, test, n_lag, n_seq):
3     forecasts = list()
4     for i in range(len(test)):
5         X, y = test[i, 0:n_lag], test[i, n_lag:]
6         # make forecast
7         forecast = persistence(X[-1], n_seq)
8         # store the forecast
9         forecasts.append(forecast)
10    return forecasts
```

We can call this function as follows:

```
1 forecasts = make_forecasts(train, test, 1, 3)
```

Evaluate Forecasts

The final step is to evaluate the forecasts.

We can do that by calculating the RMSE for each time step of the multi-step forecast, in this case giving us 3 RMSE scores. The function below, *evaluate_forecasts()*, calculates and prints the RMSE for each forecasted time step.

```
1 # evaluate the RMSE for each forecast time step
2 def evaluate_forecasts(test, forecasts, n_lag, n_seq):
3     for i in range(n_seq):
4         actual = test[:,(n_lag+i)]
```

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

```

5     predicted = [forecast[i] for forecast in forecasts]
6     rmse = sqrt(mean_squared_error(actual, predicted))
7     print('t+%d RMSE: %f' % ((i+1), rmse))

```

We can call it as follows:

```
1 evaluate_forecasts(test, forecasts, 1, 3)
```

It is also helpful to plot the forecasts in the context of the original dataset to get an idea of how the RMSE scores relate to the problem in context.

We can first plot the entire Shampoo dataset, then plot each forecast as a red line. The function `plot_forecasts()` below will create and show this plot.

```

1 # plot the forecasts in the context of the original dataset
2 def plot_forecasts(series, forecasts, n_test):
3     # plot the entire dataset in blue
4     pyplot.plot(series.values)
5     # plot the forecasts in red
6     for i in range(len(forecasts)):
7         off_s = len(series) - n_test + i
8         off_e = off_s + len(forecasts[i])
9         xaxis = [x for x in range(off_s, off_e)]
10        pyplot.plot(xaxis, forecasts[i], color='red')
11    # show the plot
12    pyplot.show()

```

We can call the function as follows. Note that the number of observations held back on the test set is supervised learning input/output patterns as was used above.

```

1 # plot forecasts
2 plot_forecasts(series, forecasts, 12)

```

We can make the plot better by connecting the persisted forecast to the actual persisted value in the

This will require adding the last observed value to the front of the forecast. Below is an updated version of the `plot_forecasts()` function with this improvement.

```

1 # plot the forecasts in the context of the original dataset
2 def plot_forecasts(series, forecasts, n_test):
3     # plot the entire dataset in blue
4     pyplot.plot(series.values)
5     # plot the forecasts in red
6     for i in range(len(forecasts)):
7         off_s = len(series) - 12 + i - 1

```

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

```

8     off_e = off_s + len(forecasts[i]) + 1
9     xaxis = [x for x in range(off_s, off_e)]
10    yaxis = [series.values[off_s]] + forecasts[i]
11    pyplot.plot(xaxis, yaxis, color='red')
12    # show the plot
13    pyplot.show()

```

Complete Example

We can put all of these pieces together.

The complete code example for the multi-step persistence forecast is listed below.

```

1  from pandas import DataFrame
2  from pandas import concat
3  from pandas import read_csv
4  from pandas import datetime
5  from sklearn.metrics import mean_squared_error
6  from math import sqrt
7  from matplotlib import pyplot
8
9  # date-time parsing function for loading the dataset
10 def parser(x):
11     return datetime.strptime('190'+x, '%Y-%m')
12
13 # convert time series into supervised learning problem
14 def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
15     n_vars = 1 if type(data) is list else data.shape[1]
16     df = DataFrame(data)
17     cols, names = list(), list()
18     # input sequence (t-n, ... t-1)
19     for i in range(n_in, 0, -1):
20         cols.append(df.shift(i))
21         names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
22     # forecast sequence (t, t+1, ... t+n)
23     for i in range(0, n_out):
24         cols.append(df.shift(-i))
25         if i == 0:
26             names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
27         else:
28             names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
29     # put it all together
30     agg = concat(cols, axis=1)
31     agg.columns = names
32     # drop rows with NaN values

```

Get Your Start in Machine Learning ×

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

```

33     if dropnan:
34         agg.dropna(inplace=True)
35     return agg
36
37 # transform series into train and test sets for supervised learning
38 def prepare_data(series, n_test, n_lag, n_seq):
39     # extract raw values
40     raw_values = series.values
41     raw_values = raw_values.reshape(len(raw_values), 1)
42     # transform into supervised learning problem X, y
43     supervised = series_to_supervised(raw_values, n_lag, n_seq)
44     supervised_values = supervised.values
45     # split into train and test sets
46     train, test = supervised_values[0:-n_test], supervised_values[-n_test:]
47     return train, test
48
49 # make a persistence forecast
50 def persistence(last_ob, n_seq):
51     return [last_ob for i in range(n_seq)]
52
53 # evaluate the persistence model
54 def make_forecasts(train, test, n_lag, n_seq):
55     forecasts = list()
56     for i in range(len(test)):
57         X, y = test[i, 0:n_lag], test[i, n_lag:]
58         # make forecast
59         forecast = persistence(X[-1], n_seq)
60         # store the forecast
61         forecasts.append(forecast)
62     return forecasts
63
64 # evaluate the RMSE for each forecast time step
65 def evaluate_forecasts(test, forecasts, n_lag, n_seq):
66     for i in range(n_seq):
67         actual = test[:,(n_lag+i)]
68         predicted = [forecast[i] for forecast in forecasts]
69         rmse = sqrt(mean_squared_error(actual, predicted))
70         print('t+%d RMSE: %f' % ((i+1), rmse))
71
72 # plot the forecasts in the context of the original dataset
73 def plot_forecasts(series, forecasts, n_test):
74     # plot the entire dataset in blue
75     pyplot.plot(series.values)
76     # plot the forecasts in red
77     for i in range(len(forecasts)):
78         off_s = len(series) - n_test + i - 1
79         off_e = off_s + len(forecasts[i]) + 1

```

Get Your Start in Machine Learning ×

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

```

80     xaxis = [x for x in range(off_s, off_e)]
81     yaxis = [series.values[off_s]] + forecasts[i]
82     pyplot.plot(xaxis, yaxis, color='red')
83     # show the plot
84     pyplot.show()
85
86 # load dataset
87 series = read_csv('shampoo-sales.csv', header=0, parse_dates=[0], index_col=0, squeeze=True, date_parser=parser)
88 # configure
89 n_lag = 1
90 n_seq = 3
91 n_test = 10
92 # prepare data
93 train, test = prepare_data(series, n_test, n_lag, n_seq)
94 # make forecasts
95 forecasts = make_forecasts(train, test, n_lag, n_seq)
96 # evaluate forecasts
97 evaluate_forecasts(test, forecasts, n_lag, n_seq)
98 # plot forecasts
99 plot_forecasts(series, forecasts, n_test+2)

```

Running the example first prints the RMSE for each of the forecasted time steps.

This gives us a baseline of performance on each time step that we would expect the LSTM to output.

```

1 t+1 RMSE: 144.535304
2 t+2 RMSE: 86.479905
3 t+3 RMSE: 121.149168

```

The plot of the original time series with the multi-step persistence forecasts is also created. The lines show the original time series and the forecast.

This context shows how naive the persistence forecasts actually are.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning



Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**. Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Multi-Step LSTM Network

In this section, we will use the persistence example as a starting point and look at the changes needed to fit an LSTM to the training data and make multi-step forecasts for the test dataset.

Prepare Data

The data must be prepared before we can use it to train an LSTM.

Get Your Start in Machine Learning

Specifically, two additional changes are required:

1. **Stationary.** The data shows an increasing trend that must be removed by differencing.
2. **Scale.** The scale of the data must be reduced to values between -1 and 1, the activation function of the LSTM units.

We can introduce a function to make the data stationary called *difference()*. This will transform the series of values into a series of differences, a simpler representation to work with.

```
1 # create a differenced series
2 def difference(dataset, interval=1):
3     diff = list()
4     for i in range(interval, len(dataset)):
5         value = dataset[i] - dataset[i - interval]
6         diff.append(value)
7     return Series(diff)
```

We can use the *MinMaxScaler* from the sklearn library to scale the data.

Putting this together, we can update the *prepare_data()* function to first difference the data and rescale the learning problem and train test sets as we did before with the persistence example.

The function now returns a scaler in addition to the train and test datasets.

```
1 # transform series into train and test sets for supervised learning
2 def prepare_data(series, n_test, n_lag, n_seq):
3     # extract raw values
4     raw_values = series.values
5     # transform data to be stationary
6     diff_series = difference(raw_values, 1)
7     diff_values = diff_series.values
8     diff_values = diff_values.reshape(len(diff_values), 1)
9     # rescale values to -1, 1
10    scaler = MinMaxScaler(feature_range=(-1, 1))
11    scaled_values = scaler.fit_transform(diff_values)
12    scaled_values = scaled_values.reshape(len(scaled_values), 1)
13    # transform into supervised learning problem X, y
14    supervised = series_to_supervised(scaled_values, n_lag, n_seq)
15    supervised_values = supervised.values
16    # split into train and test sets
17    train, test = supervised_values[0:-n_test], supervised_values[-n_test:]
18    return scaler, train, test
```

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

We can call this function as follows:

```
1 # prepare data
2 scaler, train, test = prepare_data(series, n_test, n_lag, n_seq)
```

Fit LSTM Network

Next, we need to fit an LSTM network model to the training data.

This first requires that the training dataset be transformed from a 2D array [*samples, features*] to a 3D array [*samples, timesteps, features*]. We will fix time steps at 1, so this change is straightforward.

Next, we need to design an LSTM network. We will use a simple structure with 1 hidden layer with 1 LSTM unit, then an output layer with linear activation and 3 output values. The network will use a mean squared error loss function and the efficient ADAM optimizer.

The LSTM is stateful; this means that we have to manually reset the state of the network at the end of each epoch. We will train for 1500 epochs.

The same batch size must be used for training and prediction, and we require predictions to be made on a batch basis. This means that a batch size of 1 must be used. A batch size of 1 is also called online learning as the network will learn from one training pattern (as opposed to mini batch or batch updates).

We can put all of this together in a function called *fit_lstm()*. The function takes a number of key parameters and the function returns a fit LSTM model ready for forecasting.

```
1 # fit an LSTM network to training data
2 def fit_lstm(train, n_lag, n_seq, n_batch, nb_epoch, n_neurons):
3     # reshape training into [samples, timesteps, features]
4     X, y = train[:, 0:n_lag], train[:, n_lag:]
5     X = X.reshape(X.shape[0], 1, X.shape[1])
6     # design network
7     model = Sequential()
8     model.add(LSTM(n_neurons, batch_input_shape=(n_batch, X.shape[1], X.shape[2]), stateful=True))
9     model.add(Dense(y.shape[1]))
10    model.compile(loss='mean_squared_error', optimizer='adam')
11    # fit network
12    for i in range(nb_epoch):
13        model.fit(X, y, epochs=1, batch_size=n_batch, verbose=0, shuffle=False)
14        model.reset_states()
```

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

```
15 return model
```

The function can be called as follows:

```
1 # fit model
2 model = fit_lstm(train, 1, 3, 1, 1500, 1)
```

The configuration of the network was not tuned; try different parameters if you like.

Report your findings in the comments below. I'd love to see what you can get.

Make LSTM Forecasts

The next step is to use the fit LSTM network to make forecasts.

A single forecast can be made with the fit LSTM network by calling `model.predict()`. Again, the data must be in the format of `[samples, timesteps, features]`.

We can wrap this up into a function called `forecast_lstm()`.

```
1 # make one forecast with an LSTM,
2 def forecast_lstm(model, X, n_batch):
3     # reshape input pattern to [samples, timesteps, features]
4     X = X.reshape(1, 1, len(X))
5     # make forecast
6     forecast = model.predict(X, batch_size=n_batch)
7     # convert to array
8     return [x for x in forecast[0, :]]
```

We can call this function from the `make_forecasts()` function and update it to accept the model as an argument.

```
1 # evaluate the persistence model
2 def make_forecasts(model, n_batch, train, test, n_lag, n_seq):
3     forecasts = list()
4     for i in range(len(test)):
5         X, y = test[i, 0:n_lag], test[i, n_lag:]
6         # make forecast
7         forecast = forecast_lstm(model, X, n_batch)
8         # store the forecast
9         forecasts.append(forecast)
10    return forecasts
```

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

This updated version of the `make_forecasts()` function can be called as follows:

```
1 # make forecasts
2 forecasts = make_forecasts(model, 1, train, test, 1, 3)
```

Invert Transforms

After the forecasts have been made, we need to invert the transforms to return the values back into the original scale.

This is needed so that we can calculate error scores and plots that are comparable with other models, like the persistence forecast above.

We can invert the scale of the forecasts directly using the `MinMaxScaler` object that offers an `inverse_transform()` function.

We can invert the differencing by adding the value of the last observation (prior months' shampoo sales) to the value down the forecast.

This is a little fiddly; we can wrap up the behavior in a function name `inverse_difference()` that takes forecast as arguments and returns the inverted forecast.

```
1 # invert differenced forecast
2 def inverse_difference(last_ob, forecast):
3     # invert first forecast
4     inverted = list()
5     inverted.append(forecast[0] + last_ob)
6     # propagate difference forecast using inverted first value
7     for i in range(1, len(forecast)):
8         inverted.append(forecast[i] + inverted[i-1])
9     return inverted
```

Putting this together, we can create an `inverse_transform()` function that works through each forecast, inverting the scale and differencing the

```
1 # inverse data transform on forecasts
2 def inverse_transform(series, forecasts, scaler, n_test):
3     inverted = list()
4     for i in range(len(forecasts)):
5         # create array from forecast
6         forecast = array(forecasts[i])
7         forecast = forecast.reshape(1, len(forecast))
8         # invert scaling
```

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

```

9     inv_scale = scaler.inverse_transform(forecast)
10    inv_scale = inv_scale[0, :]
11    # invert differencing
12    index = len(series) - n_test + i - 1
13    last_ob = series.values[index]
14    inv_diff = inverse_difference(last_ob, inv_scale)
15    # store
16    inverted.append(inv_diff)
17    return inverted

```

We can call this function with the forecasts as follows:

```

1 # inverse transform forecasts and test
2 forecasts = inverse_transform(series, forecasts, scaler, n_test+2)

```

We can also invert the transforms on the output part test dataset so that we can correctly calculate the RMSE scores, as follows:

```

1 actual = [row[n_lag:] for row in test]
2 actual = inverse_transform(series, actual, scaler, n_test+2)

```

We can also simplify the calculation of RMSE scores to expect the test data to only contain the output

```

1 def evaluate_forecasts(test, forecasts, n_lag, n_seq):
2     for i in range(n_seq):
3         actual = [row[i] for row in test]
4         predicted = [forecast[i] for forecast in forecasts]
5         rmse = sqrt(mean_squared_error(actual, predicted))
6         print('t+%d RMSE: %f' % ((i+1), rmse))

```

Complete Example

We can tie all of these pieces together and fit an LSTM network to the multi-step time series forecasts

The complete code listing is provided below.

```

1 from pandas import DataFrame
2 from pandas import Series
3 from pandas import concat
4 from pandas import read_csv
5 from pandas import datetime
6 from sklearn.metrics import mean_squared_error
7 from sklearn.preprocessing import MinMaxScaler
8 from keras.models import Sequential

```

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

```
9 from keras.layers import Dense
10 from keras.layers import LSTM
11 from math import sqrt
12 from matplotlib import pyplot
13 from numpy import array
14
15 # date-time parsing function for loading the dataset
16 def parser(x):
17     return datetime.strptime('190'+x, '%Y-%m')
18
19 # convert time series into supervised learning problem
20 def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
21     n_vars = 1 if type(data) is list else data.shape[1]
22     df = DataFrame(data)
23     cols, names = list(), list()
24     # input sequence (t-n, ... t-1)
25     for i in range(n_in, 0, -1):
26         cols.append(df.shift(i))
27         names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
28     # forecast sequence (t, t+1, ... t+n)
29     for i in range(0, n_out):
30         cols.append(df.shift(-i))
31         if i == 0:
32             names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
33         else:
34             names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
35     # put it all together
36     agg = concat(cols, axis=1)
37     agg.columns = names
38     # drop rows with NaN values
39     if dropnan:
40         agg.dropna(inplace=True)
41     return agg
42
43 # create a differenced series
44 def difference(dataset, interval=1):
45     diff = list()
46     for i in range(interval, len(dataset)):
47         value = dataset[i] - dataset[i - interval]
48         diff.append(value)
49     return Series(diff)
50
51 # transform series into train and test sets for supervised learning
52 def prepare_data(series, n_test, n_lag, n_seq):
53     # extract raw values
54     raw_values = series.values
55     # transform data to be stationary
```

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

```

56 diff_series = difference(raw_values, 1)
57 diff_values = diff_series.values
58 diff_values = diff_values.reshape(len(diff_values), 1)
59 # rescale values to -1, 1
60 scaler = MinMaxScaler(feature_range=(-1, 1))
61 scaled_values = scaler.fit_transform(diff_values)
62 scaled_values = scaled_values.reshape(len(scaled_values), 1)
63 # transform into supervised learning problem X, y
64 supervised = series_to_supervised(scaled_values, n_lag, n_seq)
65 supervised_values = supervised.values
66 # split into train and test sets
67 train, test = supervised_values[0:-n_test], supervised_values[-n_test:]
68 return scaler, train, test
69
70 # fit an LSTM network to training data
71 def fit_lstm(train, n_lag, n_seq, n_batch, nb_epoch, n_neurons):
72     # reshape training into [samples, timesteps, features]
73     X, y = train[:, 0:n_lag], train[:, n_lag:]
74     X = X.reshape(X.shape[0], 1, X.shape[1])
75     # design network
76     model = Sequential()
77     model.add(LSTM(n_neurons, batch_input_shape=(n_batch, X.shape[1], X.shape[2]), stateful=True))
78     model.add(Dense(y.shape[1]))
79     model.compile(loss='mean_squared_error', optimizer='adam')
80     # fit network
81     for i in range(nb_epoch):
82         model.fit(X, y, epochs=1, batch_size=n_batch, verbose=0, shuffle=False)
83         model.reset_states()
84     return model
85
86 # make one forecast with an LSTM,
87 def forecast_lstm(model, X, n_batch):
88     # reshape input pattern to [samples, timesteps, features]
89     X = X.reshape(1, 1, len(X))
90     # make forecast
91     forecast = model.predict(X, batch_size=n_batch)
92     # convert to array
93     return [x for x in forecast[0, :]]
94
95 # evaluate the persistence model
96 def make_forecasts(model, n_batch, train, test, n_lag, n_seq):
97     forecasts = list()
98     for i in range(len(test)):
99         X, y = test[i, 0:n_lag], test[i, n_lag:]
100         # make forecast
101         forecast = forecast_lstm(model, X, n_batch)
102         # store the forecast

```

Get Your Start in Machine Learning ×

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

```

103     forecasts.append(forecast)
104     return forecasts
105
106 # invert differenced forecast
107 def inverse_difference(last_ob, forecast):
108     # invert first forecast
109     inverted = list()
110     inverted.append(forecast[0] + last_ob)
111     # propagate difference forecast using inverted first value
112     for i in range(1, len(forecast)):
113         inverted.append(forecast[i] + inverted[i-1])
114     return inverted
115
116 # inverse data transform on forecasts
117 def inverse_transform(series, forecasts, scaler, n_test):
118     inverted = list()
119     for i in range(len(forecasts)):
120         # create array from forecast
121         forecast = array(forecasts[i])
122         forecast = forecast.reshape(1, len(forecast))
123         # invert scaling
124         inv_scale = scaler.inverse_transform(forecast)
125         inv_scale = inv_scale[0, :]
126         # invert differencing
127         index = len(series) - n_test + i - 1
128         last_ob = series.values[index]
129         inv_diff = inverse_difference(last_ob, inv_scale)
130         # store
131         inverted.append(inv_diff)
132     return inverted
133
134 # evaluate the RMSE for each forecast time step
135 def evaluate_forecasts(test, forecasts, n_lag, n_seq):
136     for i in range(n_seq):
137         actual = [row[i] for row in test]
138         predicted = [forecast[i] for forecast in forecasts]
139         rmse = sqrt(mean_squared_error(actual, predicted))
140         print('t+%d RMSE: %f' % ((i+1), rmse))
141
142 # plot the forecasts in the context of the original dataset
143 def plot_forecasts(series, forecasts, n_test):
144     # plot the entire dataset in blue
145     pyplot.plot(series.values)
146     # plot the forecasts in red
147     for i in range(len(forecasts)):
148         off_s = len(series) - n_test + i - 1
149         off_e = off_s + len(forecasts[i]) + 1

```

Get Your Start in Machine Learning ×

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning


```

150     xaxis = [x for x in range(off_s, off_e)]
151     yaxis = [series.values[off_s]] + forecasts[i]
152     pyplot.plot(xaxis, yaxis, color='red')
153     # show the plot
154     pyplot.show()
155
156 # load dataset
157 series = read_csv('shampoo-sales.csv', header=0, parse_dates=[0], index_col=0, squeeze=True, date_parser=parser)
158 # configure
159 n_lag = 1
160 n_seq = 3
161 n_test = 10
162 n_epochs = 1500
163 n_batch = 1
164 n_neurons = 1
165 # prepare data
166 scaler, train, test = prepare_data(series, n_test, n_lag, n_seq)
167 # fit model
168 model = fit_lstm(train, n_lag, n_seq, n_batch, n_epochs, n_neurons)
169 # make forecasts
170 forecasts = make_forecasts(model, n_batch, train, test, n_lag, n_seq)
171 # inverse transform forecasts and test
172 forecasts = inverse_transform(series, forecasts, scaler, n_test+2)
173 actual = [row[n_lag:] for row in test]
174 actual = inverse_transform(series, actual, scaler, n_test+2)
175 # evaluate forecasts
176 evaluate_forecasts(actual, forecasts, n_lag, n_seq)
177 # plot forecasts
178 plot_forecasts(series, forecasts, n_test+2)

```

Running the example first prints the RMSE for each of the forecasted time steps.

We can see that the scores at each forecasted time step are better, in some cases much better, than

This shows that the configured LSTM does have skill on the problem.

It is interesting to note that the RMSE does not become progressively worse with the length of the forecast horizon, as would be expected. This is marked by the fact that the t+2 appears easier to forecast than t+1. This may be because the downward tick is easier to predict than the upward tick noted in the series (this could be confirmed with more in-depth analysis of the results).

```

1 t+1 RMSE: 95.973221
2 t+2 RMSE: 78.872348
3 t+3 RMSE: 105.613951

```

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

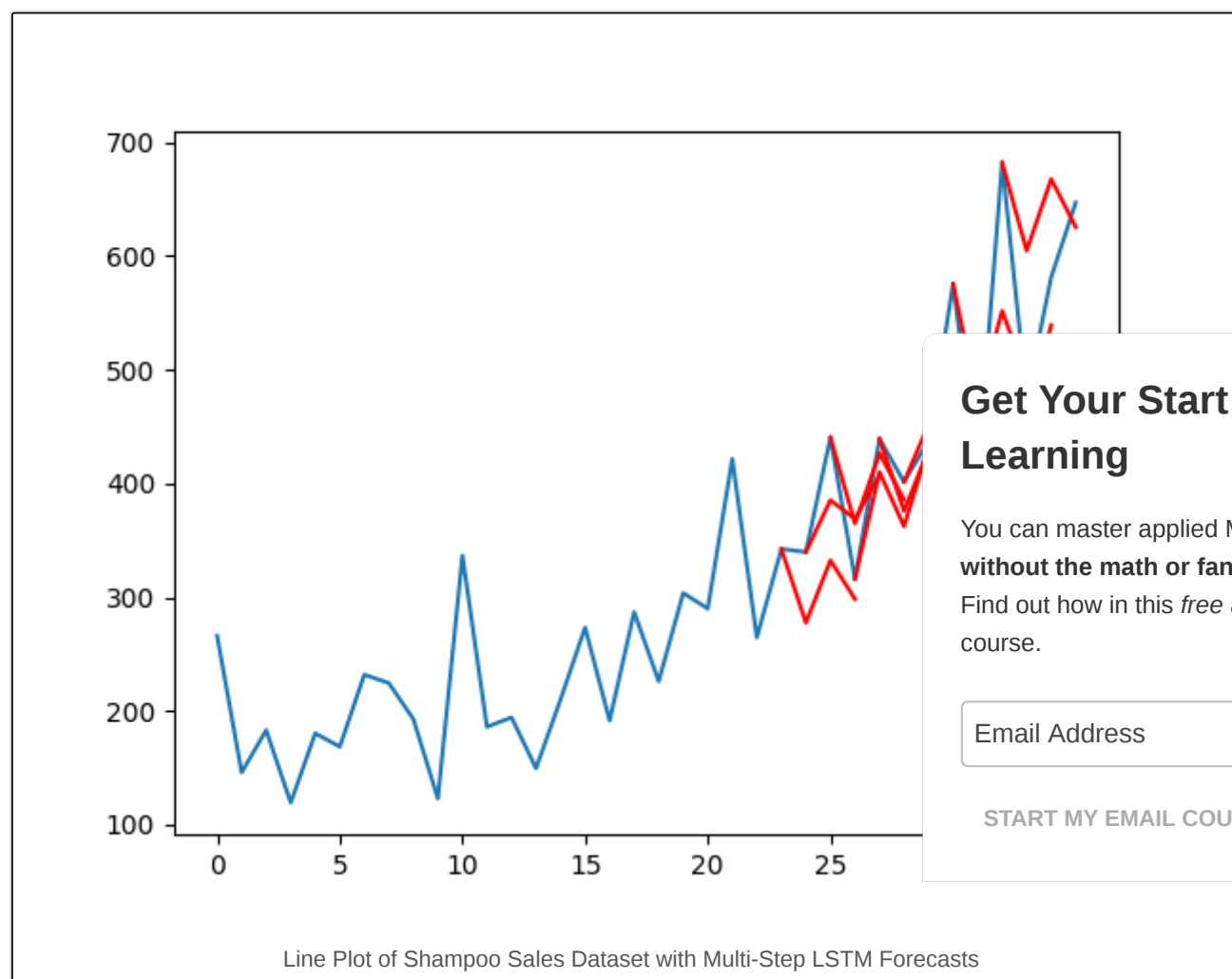
Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

A line plot of the series (blue) with the forecasts (red) is also created.

The plot shows that although the skill of the model is better, some of the forecasts are not very good and that there is plenty of room for improvement.



Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Extensions

There are some extensions you may consider if you are looking to push beyond this tutorial.

Get Your Start in Machine Learning

- **Update LSTM.** Change the example to refit or update the LSTM as new data is made available. A 10s of training epochs should be sufficient to retrain with a new observation.
- **Tune the LSTM.** Grid search some of the LSTM parameters used in the tutorial, such as number of epochs, number of neurons, and number of layers to see if you can further lift performance.
- **Seq2Seq.** Use the encoder-decoder paradigm for LSTMs to forecast each sequence to see if this offers any benefit.
- **Time Horizon.** Experiment with forecasting different time horizons and see how the behavior of the network varies at different lead times.

Did you try any of these extensions?

Share your results in the comments; I'd love to hear about it.

Summary

In this tutorial, you discovered how to develop LSTM networks for multi-step time series forecasting.

Specifically, you learned:

- How to develop a persistence model for multi-step time series forecasting.
- How to develop an LSTM network for multi-step time series forecasting.
- How to evaluate and plot the results from multi-step time series forecasting.

Do you have any questions about multi-step time series forecasting with LSTMs?

Ask your questions in the comments below and I will do my best to answer.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Develop LSTMs for Sequence Prediction Today!

Develop Your Own LSTM models in Minutes

...with just a few lines of python code

Discover how in my new Ebook:

[Long Short-Term Memory Networks with Python](#)

Get Your Start in Machine Learning

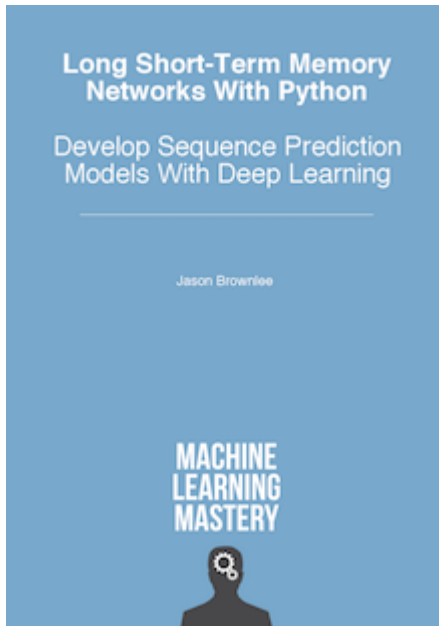
It provides **self-study tutorials** on topics like:

CNN LSTMs, Encoder-Decoder LSTMs, generative models, data preparation, making predictions and much more...

Finally Bring LSTM Recurrent Neural Networks to Your Sequence Predictions Projects

Skip the Academics. Just Results.

[Click to learn more.](#)



About Jason Brownlee

Dr. Jason Brownlee is a husband, proud father, academic researcher, author, professional developer, and entrepreneur. He is passionate about helping developers get started and get good at applied machine learning. [Learn more.](#)

[View all posts by Jason Brownlee](#) →

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)

[← How to Convert a Time Series to a Supervised Learning Problem in Python](#)

[Demonstration of Memory with a Long Short-Term Memory Network in Python >](#)

[Get Your Start in Machine Learning](#)

129 Responses to *Multi-step Time Series Forecasting with Long Short-Term Memory Networks in Python*



Masum May 10, 2017 at 6:48 am #

REPLY ↩

Thanks

you are the best

Did not had to wait for long. Asked for it in different blog few days back



Jason Brownlee May 10, 2017 at 8:53 am #

I hope you find the post useful!



Masum May 10, 2017 at 9:59 am #

I believe so. Things are getting deeper here.

Will we get recursive LSTM MODEL for multi step forecasting soon?

Will eagerly wait for that blog.

Thanks



Jason Brownlee May 11, 2017 at 8:22 am #

REPLY ↩

Maybe.

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning



Masum May 11, 2017 at 8:43 am #

Sir,

Hope to see that soon.



jvr May 17, 2017 at 1:27 am #

REPLY ↩

Thanks a lot for this post. I was trying to make this for my thesis since september, with no well results. But I'm having trouble: I'm not able to compile. Maybe you or someone who reads this is able to tell me why this happens: I'm getting the following error when running the code:

The TensorFlow library wasn't compiled to use SSE instructions, but these are available on your machine

The TensorFlow library wasn't compiled to use SSE2 instructions, but these are available on your machine

The TensorFlow library wasn't compiled to use SSE3 instructions, but these are available on your machine

The TensorFlow library wasn't compiled to use SSE4.1 instructions, but these are available on your machine

The TensorFlow library wasn't compiled to use SSE4.2 instructions, but these are available on your machine

The TensorFlow library wasn't compiled to use AVX instructions, but these are available on your machine

Obviously it has something to do with Tensorflow (I have read about this problem and I think its because i
to fix it).

Thank you in advance.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Jason Brownlee May 17, 2017 at 8:38 am #

REPLY ↩

These are warnings that you can ignore.

Get Your Start in Machine Learning



Shamsul May 17, 2017 at 9:17 pm #

REPLY ↩

Sir,

Can we say that multiple output strategy (avoiding 1.direct, 2. Recursive, 3.direct recursive hybrid strategies) have been used here ?

Am I right ?



Jason Brownlee May 18, 2017 at 8:36 am #

REPLY ↩

I think the LSTM has implemented a direct strategy.



jinhua zhang May 18, 2017 at 11:26 am #

Hi,Jason,

Your article is very useful! I have a problem, if the data series are three-dimensional data, the 2th line is data(all include the train and test data),Do they can run the" difference"and "tansform"?

Thank you very much!

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Jason Brownlee May 19, 2017 at 8:11 am #

Great question.

You may want to only make the prediction variable stationary. Consider perform three tests:

- Model as-is
- Model with output variable stationary
- Model with all variables stationary (if others are non-stationary)

Get Your Start in Machine Learning



jvr May 21, 2017 at 10:21 pm #

I have discovered how to do it by asking some people. The object series is actually a Pandas Series. It's a vector of information, with a named index. Your dataset, however, contains two fields of information, in addition to the time series index, which makes it a DataFrame. This is the reason why the tutorial code breaks with your data.

To pass your entire dataset to MinMaxScaler, just run `difference()` on both columns and pass in the transformed vectors for scaling. MinMaxScaler accepts an n-dimensional DataFrame object:

```
ncol = 2
diff_df = pd.concat([difference(df[i], 1) for i in range(1,ncol+1)], axis=1)
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_values = scaler.fit_transform(diff_df)
```

So, with this, we can use as many variables as we want. But now I have a big doubt.

When the transform or dataset into a supervised learning problem, we have a distribution in columns
<http://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/>

I mean, for a 2 variables dataset as yours, we can set, for example, this values:

```
n_lags=1
n_seq=2
```

so we will have a supervised dataset like this:

```
var1(t-1) var2(t-1) var1(t) var2 (t) var1(t+1) var2 (t+1)
```

so, if we want to train the ANN to forecast var2 (which is the target we want to predict) with the var1 we have to separate them and here is where my doubt begins.

In the part of the code:

```
def fit_lstm(train, n_lag, n_seq, n_batch, nb_epoch, n_neurons):
# reshape training into [samples, timesteps, features]
X, y = train[:, 0:n_lag], train[:, n_lag:]
X = X.reshape(X.shape[0], 1, X.shape[1])
```

I think that if we want to define X, we should use:

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning


```
X=train[:,0:n_lag*n_vars]
```

this means we are selecting this as X from the previous example:

```
var1(t-1) var2(t-1)
```

(number of lags*number of variables), so: `X=train[:,0:1*2]=train[:,0:2]`

but...

`Y=train[:,n_lag*n_vars:]` is the vector of targets?

the problem is that, on this way, we are selecting this as targets:

```
var1(t) var2(t) var1(t+1) var2(t+1)
```

so we are including var1 (which we don't have the aim to forecast, just use as input).

I would like to know if there is any solution to solve this in order to use the variable 1,2...n-1 just as input

Hope this is clear :/



jvr May 19, 2017 at 3:16 am #

Thanks for the previous clarification. I have a doubt in relation to the section "fit network" in the course graph (validation vs training) in order to see if the network is or not overfitted, but due to the "model.reset_val_loss" from the history sentence. Is there any way to solve this?

thank you in advance 😊

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



jvr May 19, 2017 at 3:45 am #

REPLY ↩

I reply to myself, if someone is also interested.

Just creating 2 lists (or 1, but I see it more clear on this way) and returning them on the function. Then, outside, just plot them. I'm sorry for the question, maybe the answer is obvious, but I'm starting on Python and I'm not a programmer.

Get Your Start in Machine Learning

```
# fit network
loss=list()
val_loss=list()
for i in range(nb_epoch):
    history=model.fit(X, y, epochs=1, batch_size=n_batch,shuffle=True, validation_split=val_split)
    eqm=history.history['loss']
    eqm_val=history.history['val_loss']
    loss.append(eqm)
    val_loss.append(eqm_val)
    model.reset_states()

return model,loss,val_loss

# fit model
model,loss,val_loss=fit_lstm(train, n_lag, n_seq, n_batch, n_epochs, n_neurons)

pyplot.figure()
pyplot.plot(loss)
pyplot.plot(val_loss)
pyplot.title('cross validation')
pyplot.ylabel('MSE')
pyplot.xlabel('epoch')
pyplot.legend(['training', 'test'], loc='upper left')
pyplot.show()
```



Jason Brownlee May 19, 2017 at 8:23 am #

Nice to see you got there jvr, well done.



Jason Brownlee May 19, 2017 at 8:22 am #

History is returned when calling model.fit().

Get Your Start in Machine Learning ×

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

REPLY

REPLY ↩

Get Your Start in Machine Learning

We are only fitting one epoch at a time, so you can retrieve and accumulate performance each epoch in the epoch loop then do something with the data (save/graph/return it) at the end of the loop.

Does that help?



jvr May 19, 2017 at 9:17 pm #

REPLY ↩

It does help, thank you.

Now I'm trying to find a way to make the training process faster and reduce RMSE, but it's pretty difficult (the idea is to make results better than in the NARx model implemented in the Matlab Neural Toolbox, but results and computational time are hard to overcome).



Jason Brownlee May 20, 2017 at 5:37 am #

LSTMs often need to be trained longer than you think and can greatly benefit from



DJ June 2, 2017 at 1:42 am #

Hi,

Thanks for the great tutorial, I'm wondering if you can help me clarify the reason you have

`model.reset_states()`

(line 83)

when fitting the model, I was able to achieve similar results without the line as well.

Thanks!

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning



Jason Brownlee June 2, 2017 at 1:02 pm #

REPLY ↩

It clears the internal state of the LSTM.



anurag August 30, 2017 at 3:41 pm #

REPLY ↩

I have tried experimenting with and without `model.reset_states()`, using some other dataset. I am doing multistep prediction for 6-10 steps, I am able to get better results without `model.reset_states()`. Am i doing something wrong, or it completely depends on dataset to dataset. Thanks in advance.



Jason Brownlee August 30, 2017 at 4:20 pm #

It completely depends on the dataset and the model.



anurag August 31, 2017 at 6:42 pm #

Thank you so much. 😊



DJ June 2, 2017 at 4:11 pm #

REPLY ↩

Thanks for the quick reply Jason :-). I've seen other places where reset is done by using `callbacks` parameter in `model.fit`.

```
class ResetStatesCallback(Callback):
```

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree**. Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

```
def __init__(self):
    self.counter = 0

def on_batch_begin(self, batch, logs={}):
    if self.counter % max_len == 0:
        self.model.reset_states()
    self.counter += 1
```

Then the callback is used by as follows:

```
model.fit(X, y, epochs=1, batch_size=1, verbose=2,
        shuffle=False, callbacks=[ResetStatesCallback()])
```

The ResetStatesCallback snippet was obtained from:

<http://philipperemy.github.io/keras-stateful-lstm/>

Please let me know what you think.

Thanks!



Jason Brownlee June 3, 2017 at 7:21 am #

Yes, there are many ways to implement the reset. Use what works best for your application



QQ June 2, 2017 at 5:00 pm #

Hi Jason, greate post, and I have some questions:

1. in your fit_lstm function, you reset each epoch state, why?
2. why you iterate each epoch by yourself, instead of using model.fit(X, y, epochs)

thx Jason

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

REPLY ↩

Get Your Start in Machine Learning

```
# fit an LSTM network to training data
def fit_lstm(train, n_lag, n_seq, n_batch, nb_epoch, n_neurons):
    # reshape training into [samples, timesteps, features]
    X, y = train[:, 0:n_lag], train[:, n_lag:]
    X = X.reshape(X.shape[0], 1, X.shape[1])
    # design network
    model = Sequential()
    model.add(LSTM(n_neurons, batch_input_shape=(n_batch, X.shape[1], X.shape[2]), stateful=True))
    model.add(Dense(y.shape[1]))
    model.compile(loss='mean_squared_error', optimizer='adam')
    # fit network
    for i in range(nb_epoch):
        model.fit(X, y, epochs=1, batch_size=n_batch, verbose=0, shuffle=False)
    model.reset_states()
    return model
```



Jason Brownlee June 3, 2017 at 7:23 am #

The end of the epoch is the end of the sequence and the internal state should not carry over

I run the epochs manually to give fine grained control over when resets occur (by default they occur



J June 7, 2017 at 12:48 am #

I'd like to clarify line 99 in the LSTM example:

```
— plot_forecasts(series, forecasts, n_test+2)
```

Is the $n_test + 2 == n_test + n_lag - n_seq$?

Thanks,
J

Get Your Start in Machine Learning ×

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning



jvr June 15, 2017 at 11:49 pm #

REPLY ↩

I'd also like to know why using `n_test + 2`



M August 8, 2017 at 3:07 am #

REPLY ↩

I thought it should be `n_test + 2 == n_test + n_seq - 1` (regardless of `n_seq`). It would be great if someone could clarify that.



Mrtn October 4, 2017 at 8:36 pm #

M, you are right. Otherwise the RMS is incorrectly calculated and plotting is not aligned



Kao June 10, 2017 at 5:46 pm #

Hi jason,

When I applied your code into a 22-year daily time series, I find out that the LSTM forecast result is similar to a horizontal bar. I'm sure I did not mess those two methods, I wonder what cause this?

My key configure as follows:

`n_lag = 1`

`n_seq = 3`

`n_test = 365*3`

and my series length is 8035.

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Jason Brownlee June 11, 2017 at 8:21 am #

Get Your Start in Machine Learning



You will need to tune the model to your problem.



Kao June 25, 2017 at 6:55 pm #

REPLY ↩

Thanks to your tutorial, I've been tuning the parameters such as numbers of epochs and neurons these days. However, I noticed that you mentioned the grid search method to get appropriate parameters, could you please explain how to implement it into LSTM? I'm confused about your examples on some other tutorial which has a model class, seems unfamiliar to me.



Jason Brownlee June 26, 2017 at 6:07 am #

See this example on how to grid search with LSTMs manually:
<http://machinelearningmastery.com/tune-lstm-hyperparameters-keras-time-series-forecasting/>



Kao June 28, 2017 at 1:25 am #

Thanks, I've just finished one test. What does it mean if error oscillates violently diminishing? Can I tune the model better, or LSTM is incapable of this time series?



Jason Brownlee June 28, 2017 at 6:28 am #

You may need a larger model (more layers and or more neurons).



MM June 13, 2017 at 6:44 am #

Jason,

REPLY ↩

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

Thank you for these tutorials. These are the best tutorials on the web. One question: what is the best way to forecast the last two values?

Thank you



Jason Brownlee June 13, 2017 at 8:31 am #

REPLY ↩

Thanks MM.

No one can tell you the “best” way to do anything in applied machine learning, you must discover it through trial and error on your specific problem.



MM June 13, 2017 at 9:29 am #

Jason,

Understood. Let me re-phrase the question. In a practical application, one would be interested in a dataset, “3-12”. How would you suggest doing that?



Jason Brownlee June 14, 2017 at 8:41 am #

Fit your model to all of the data then call predict() passing whatever lag inputs your

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



MM June 13, 2017 at 10:24 am #

REPLY ↩

Jason,

Should the line that starts the offset point in plot_forecasts() be

```
off_s = len(series) - n_test + i + 1
```

not

Get Your Start in Machine Learning

```
off_s = len(series) - n_test + i - 1
```



Michael June 21, 2017 at 4:03 am #

REPLY ↩

Hi Jason,

Thanks for your excellent tutorials!

I have followed a couple of your articles about LSTM and did learn a lot, but here is a question in my mind: can I introduce some interference elements in the model? For example for shampoo sale problem, there may be some data about holiday sales, or sales data after an incident happens. If I want to make prediction for sales after those incidents, what can I do?

What's more, I noticed that you will parse date/time with a parser, but you did not really introduce time feature for prediction for next Monday or next January, how can I feed time feature?

Thanks!



Jason Brownlee June 21, 2017 at 8:18 am #

Yes, see this post for ideas on adding additional features:

<http://machinelearningmastery.com/basic-feature-engineering-time-series-data-python/>



Michael June 22, 2017 at 5:53 pm #

Thanks for clarification.

I have two more specific questions:

1) In `inverse_transform`, why `index = len(series) - n_test + i - 1`?

2) In `fit_lstm`, you said "reshape training into [samples, timesteps, features]", but I think the code in line 74 is a little different from your format:

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

```
73 X, y = train[:, 0:n_lag], train[:, n_lag:]  
74 X = X.reshape(X.shape[0], 1, X.shape[1])
```

In line 74, I think it should be `X = X.reshape(X.shape[0], X.shape[1], 1)`



Jason Brownlee June 23, 2017 at 6:52 am #

REPLY ↩

Hi Michael,

Yes, the offset finds one step prior to the forecast in the original time series. I use this motif throughout the tutorial.

In the very next line I say: "We will fix time steps at 1, so this change is straightforward."



Michael June 22, 2017 at 6:01 pm #

Hi Jason,

I would like to know how to do short term and long term prediction with minimum number of models?

For example, I have a 12-step input and 12-step output model A, and a 12-step input and 1-step output model B. Is model A the next first time step than model B?

What's more, if we have 1-step input and 1-step output model, it is more error prone to long term prediction. If we have multi-step input and 1-step output model it is still more error prone long term. So how to

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Jason Brownlee June 23, 2017 at 6:53 am #

REPLY ↩

I would recommend developing and evaluating each model for the different use cases. LSTMs are quite resistant to assumptions and rules of thumb I find in practice.

Get Your Start in Machine Learning



jzx June 25, 2017 at 1:17 pm #

REPLY ↩

Hello, thanks for your tutorial

If my prediction model is three time series a, b, c, I would like to use a, b, c to predict the future a, how can I build my LSTM model.
thank you very much!



Jason Brownlee June 26, 2017 at 6:05 am #

REPLY ↩

Each of a, b, and c would be input features. Remember, the shape or dimensions of input data is [samples, timesteps, features].



Kedar June 26, 2017 at 6:03 pm #

Does stationarizing data really help the LSTM? If so, what is the intuition behind that? I mean, I know it helps for LSTM's?



Jason Brownlee June 27, 2017 at 8:27 am #

Yes in my experience, namely because it is a simpler prediction problem.

I would suggest trying a few different “views” of your sequence and see what is easiest to model / get

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**. Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Michael June 28, 2017 at 5:47 pm #

REPLY ↩

Hi Jason,

I want to train a model with the following input size: [6000, 4, 2] ([samples, timestamps, features])

Get Your Start in Machine Learning

For example, I want to predict shampoo's sale in next two years. If I have other feature like economy index of every year, can I concatenate sale data and index data in the above format? So my input will be a 3d vector. How should I modify the model to train?

I always get such error: ValueError: Error when checking target: expected dense_1 to have 2 dimensions, but got array with shape (6000, 2, 2).

The error comes from this line: `model.fit(X, y, epochs=1, batch_size=n_batch, verbose=0, shuffle=False)`. Can you provide some advices? Thanks!



Jason Brownlee June 29, 2017 at 6:32 am #

REPLY ↩

Reshape your data to be [6000, 4, 2]

Update the input shape of the network to be (4,2)

Adjust the length of the output sequence you want to predict.



shamsul July 11, 2017 at 11:31 am #

sir,

To make one forecast with an LSTM, if we write

```
oneforecast = forecast_lstm(model, X, n_batch)
```

it says: undefined X

what should be the value of X? we know the model and n_batch value?

would you help?

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Jason Brownlee July 12, 2017 at 9:38 am #

REPLY ↩

X would be the input sequence required to make a prediction, e.g. lag obs.

Get Your Start in Machine Learning



masum July 12, 2017 at 8:06 am #

REPLY ↩

sir,

what if I want to tell the model to learn from train data (23 samples here) and want to forecast only 3 steps forward (Jan, Feb, Mar). I want to avoid persistence model in this case and only require 3 step direct strategy. hope you got that.

any help would be grateful.

tarin (past data)= forecast (Jan, Feb, Mar)



Jason Brownlee July 12, 2017 at 9:54 am #

Perhaps I misunderstand, but this is the model presented in the tutorial. It predicts 3 time st



masum July 12, 2017 at 11:00 am #

evaluate the persistence model

```
def make_forecasts(model, n_batch, train, test, n_lag, n_seq):
    forecasts = list()
    for i in range(len(test)):
        X, y = test[i, 0:n_lag], test[i, n_lag:]
        # make forecast
        forecast = forecast_lstm(model, X, n_batch)
        # store the forecast
        forecasts.append(forecast)
    return forecasts
```

here if i would like to make only one forecast for 3 steps (jan,feb,march) what i have to change. i do not need the rest of the month(april, may, june, july,aug,.....dec). one predictions or forecast for 3 steps.

hope you got me

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning



Jason Brownlee July 13, 2017 at 9:47 am #

REPLY ↩

Pass in only what is required to make the prediction for those 3 months.



masum July 13, 2017 at 10:16 am #

sir,

will be kind enough to simplify a little bit more.

I did not get it.



Devakar Kumar Verma July 24, 2017 at 4:23 am #

I am getting an error while parsing the date at time of loading the data from csv file.

The error is:

ValueError: time data '1901-Jan' does not match format '%Y-%m'

Anyone please help me to resolve this issue.

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Jason Brownlee July 24, 2017 at 6:56 am #

REPLY ↩

I'm sorry to hear that. Confirm you have copied the code exactly and the data file does not have any extra footer information.



p July 30, 2017 at 8:05 pm #

REPLY ↩

Get Your Start in Machine Learning

hi

I have so this problem

i have downloaded the dataset from the link in the text

i think this error has occurred because the data of our csv file is not in correct format!

can anyone give us the dataset plz???



Jason Brownlee July 31, 2017 at 8:15 am #

REPLY ↩

Here is the raw data ready to go:

```
1 "Month", "Sales"
2 "1-01", 266.0
3 "1-02", 145.9
4 "1-03", 183.1
5 "1-04", 119.3
6 "1-05", 180.3
7 "1-06", 168.5
8 "1-07", 231.8
9 "1-08", 224.5
10 "1-09", 192.8
11 "1-10", 122.9
12 "1-11", 336.5
13 "1-12", 185.9
14 "2-01", 194.3
15 "2-02", 149.5
16 "2-03", 210.1
17 "2-04", 273.3
18 "2-05", 191.4
19 "2-06", 287.0
20 "2-07", 226.0
21 "2-08", 303.6
22 "2-09", 289.9
23 "2-10", 421.6
24 "2-11", 264.5
25 "2-12", 342.3
26 "3-01", 339.7
27 "3-02", 440.4
28 "3-03", 315.9
29 "3-04", 439.3
30 "3-05", 401.3
31 "3-06", 437.4
```

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning


```
32 "3-07",575.5
33 "3-08",407.6
34 "3-09",682.0
35 "3-10",475.3
36 "3-11",581.3
37 "3-12",646.9
```



Dongchan October 9, 2017 at 9:26 am #

REPLY ↩

Sir,

I have the same issue. How can I fix the parser to resolve this error?



Devakar Kumar Verma July 24, 2017 at 2:34 pm #

@Jason,

Data file doesn't have any footer and i had simply copy paste the code but dateparser throwing the error



Jason Brownlee July 25, 2017 at 9:27 am #

Sorry, I don't have any good ideas. It may be a Python environment issue?



Josep July 31, 2017 at 8:15 pm #

REPLY ↩

Hi Jason,

Great explanation again. I have a doubt about this piece of code:

```
# evaluate the persistence model
```

```
def make_forecasts(model, n_batch, train, test, n_lag, n_seq):
```

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

```
forecasts = list()
for i in range(len(test)):
    X, y = test[i, 0:n_lag], test[i, n_lag:]
    # make forecast
    forecast = forecast_lstm(model, X, n_batch)
    # store the forecast
    forecasts.append(forecast)
return forecasts
```

Why do you pass the parameter “n_seq” to the function if it has no use inside the function?



Jason Brownlee August 1, 2017 at 7:59 am #

Good point, thanks.



Nara August 1, 2017 at 10:12 pm #

Hi,

How would I go about forecasting for a complete month. (Assuming I have daily data).
Assuming I have around 5 years data 1.8k data points to train.

I would like to use one year old data to forecast for the whole of next month?

To do this should I change the way this model is trained?

Is my understanding correct that this model tries to predict the next value by only using current value?

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**
Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Jason Brownlee August 2, 2017 at 7:50 am #

Yes, frame the data so that it predicts a month, then train the model.

The model can take as input whatever you wish, e.g. a sequence of the last month or year.

Get Your Start in Machine Learning



Nara August 3, 2017 at 3:12 am #

REPLY ↩

Hey, thanks for the reply.

This post really helped me.

Now the next question is how do we enhance this to consider exogenous variables while forecasting?

If I simply add exogenous variable values at this step:

`train, test = supervised_values[0:-n_test], supervised_values[-n_test:]`, (and obviously make appropriately changes to `batch_input_shape` in model fit.)

Would it help improve predictions?

What is the correct way of adding independent variables.

I have gone through this post of yours.

<http://machinelearningmastery.com/basic-feature-engineering-time-series-data-python/>

It was helpful but how to do this using neural networks that has LSTM?

Can you please point me in the right direction?



Jason Brownlee August 3, 2017 at 6:55 am #

Additional features can be provided directly to the model as new features.

See this post on framing the problem, then reshape the results:

<http://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/>

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Kiran August 4, 2017 at 2:09 pm #

REPLY ↩

Hi Jason, thanks for writing up such detailed explanations.

I am using an LSTM layer for a time series prediction problem.

Everything works fine except for when I try to use the `inverse_transform` to undo the scaling of my data. I get the following error:

`ValueError: Input contains NaN, infinity or a value too large for dtype('float64').`

Get Your Start in Machine Learning

Not really sure how I can get past this problem. Could you please help me with this ?



Jason Brownlee August 4, 2017 at 3:45 pm #

REPLY ↩

It looks like you are trying to perform an inverse transform on NaN values.

Perhaps try some print statements to help track down where the NaN values are coming from.



Kiran August 5, 2017 at 12:01 pm #

REPLY ↩

Thank you for the reply. Yes, there are some NaN values in my predictions. Does that in



Jason Brownlee August 6, 2017 at 7:36 am #

Your model might be receiving NaN as input, check that.

It may be making NaN predictions with good input, in which case it might have had trouble doing gradient clipping that can address this.

<https://keras.io/optimizers/>

Figure out which case it is first though.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Kiran August 14, 2017 at 11:05 pm #

Thanks ! My inputs do not have any NaN. Will check out gradient clipping.

Get Your Start in Machine Learning



Jason Brownlee August 15, 2017 at 6:37 am #

Let me know how you go Kiran.



Nara August 8, 2017 at 9:34 pm #

REPLY ↩

Hi Jason,

When I try step by step forecast. i.e. forecast 1 point and then use this back as data and forecast the next point, my predictions become constant after just 2 steps, sometimes from the beginning itself.

<https://datascience.stackexchange.com/questions/22047/time-series-forecasting-with-rnnstateful-lstm-pr>

In detail there. Can you say why this is happening? And which forecast method is usually better. Step by

Also can you comment on when can ARIMA/ linear models perform better than netowrks/RNN?

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Jason Brownlee August 9, 2017 at 6:30 am #

Using predictions as input is bad as the errors will compound. Only do this if you cannot get

If your model has a linear relationship it will be better to model it with a linear model with ARIMA, the



Nara August 11, 2017 at 10:09 pm #

REPLY ↩

But that is how ARIMA models predict right?

They do point by point forecast. And from my results ARIMA(or STL ARIMA or even XGBOOST) is doing pretty well when compared to RNN. 😞

But i haven't considered stationarity and outlier treatment and I see that RNN performs pathetically when the data is non stationary/has outliers.

Is this expected? I have read that RNN should take care of stationarity automatically?

Get Your Start in Machine Learning

Also, will our results be bad if we do first order differencing even when there is no stationarity in the data?

And as for normalization, is it possible that for some cases RNN does well without normalizing?

When is normalization usually recommended? When standard deviation is huge?



Jason Brownlee August 12, 2017 at 6:49 am #

REPLY ↩

I have found RNNs to not perform well on autoregression problems, and they do better with more data prep (e.g. removing anything systematic). See this post:

<http://machinelearningmastery.com/suitability-long-short-term-memory-networks-time-series-forecasting/>

Generally, don't difference if you don't need to, but test everything to be sure.

Standardization if the distribution is Gaussian, normalization otherwise. RNNs like LSTMs need relu.



Nara August 13, 2017 at 1:34 am #

Oh then a hybrid model using residuals from ARIMA for RNN should work well
The residuals will not have any seasonal components.(even scaling should be well taken)
Or here also do you expect MLPs to work better?



Jason Brownlee August 13, 2017 at 9:55 am #

It is hard to know for sure, I recommend using experiments to collect data to know for sure, rather than guessing.



Nights August 13, 2017 at 5:37 am #

REPLY ↩

Get Your Start in Machine Learning

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**
Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

I think there is an issue with inverse differencing while forecasting for multistep.(to deal with non stationary data)

This example is adding previously forecasted(and inverse differenced) value to the currently forecasted value.Isn't this method wrong when we have 30 points to forecast as it keeps adding up the results and hence the output will continuously increase.

Below is the output I got.

<https://ibb.co/d1oyNF>

Instead should I just add the last known real observation to all the forecasted values? I dont suppose that would work either.



Jason Brownlee August 13, 2017 at 9:58 am #

REPLY ↩

It could be an issue for long lead times, as the errors will compound.

If real obs are available to use for inverse differencing, you won't need to make a forecast for such a

Consider contrasting model skill with and without differencing, at least as a starting point.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Sandra August 14, 2017 at 5:46 pm #

Hi, thank you for your helpful tutorial.

I have a question regarding a seq to seq timeseries forecasting problem with multi-step lstm.

I have created a supervised dataset of (t-1), (t-2), (t-3)..., (t-look_back) and (t+1), (t+2), (t+3)..., (t+look_ ahead) timesteps.

We have tried your complete example code of doing a dense(look_ahead) last layer but received not so good results. This was done using both a stateful and non-stateful network.

We then tried using Dense(1) and then repeatvector(look_ahead), and we get the same (around average) value for all the look_ahead timesteps. This was done using a non-stateful network.

Then I created a stepwise prediction where look_ahead = 1 always. The prediction for t+2 is then based on the history of (t+1)(t)(t-1)... This has given me better results, but only tried for non-stateful network.

Get Your Start in Machine Learning

My questions are:

- Is it possible to use repeatvector with non-stateful networks? Or must network be stateful? Do you have any idea why my predictions are all the same value?
- What do network you recommend for this type or problem? Stateful or non stateful, seq to seq or stepwise prediction?

Thanks in advance!

Sandra



Jason Brownlee August 15, 2017 at 6:32 am #

REPLY ↩

Very nice work Sandra, thanks for sharing.

The RepeatVector is only for the Encoder-Decoder architecture to ensure that each time step in the encoding vector from the Encoder. It is not related to stateful or stateless models.

I would develop a simple MLP baseline with a vector output and challenge all LSTM architectures to LSTM and a seq2seq model. I would also try the recursive model (feed outputs as inputs for repeating)

It sounds like you're trying all the right things.

Now, with all of that being said, LSTMs may not be very good at simple autoregression problems. I can't say for sure, but I think you should try a simple autoregression. See this post:

<http://machinelearningmastery.com/suitability-long-short-term-memory-networks-time-series-forecasting/>

I hope that helps, let me know how you go.

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Oscar August 16, 2017 at 1:28 am #

REPLY ↩

Hi Jason,

Thanks for your tutorials. I'm trying to learn ML and your webpage is very useful!

I'm a bit confuse with the inverse_difference function. Specifically with the last_ob that I need to pass.

Let's say I have the following:

Get Your Start in Machine Learning

Raw Data difference scaled Forecasted values

raw_val1=.4

raw_val2=.35 -.05 -.045 [0.80048585, 0.59788215, -0.13518856]

raw_val3=.29 -.06 -.054 [0.65341175, 0.37566081, -0.14706305]

raw_val4=.28 -.01 -.009 [[0.563694, -0.09381149, 0.03976132]

When passing the last_ob to the inverse_difference function which observation do I need to pass to the function, raw_val2 or raw_val1?

My hunch is that I need to pass raw_val2. Is that correct?

Also, in your example, in the line:

```
forecasts = inverse_transform(series, forecasts, scaler, n_test+2)
```

What's the reason of this n_test+2?

Thanks in advance!

Oscar



Jaskaran August 17, 2017 at 10:57 am #

Hi Jason,

Great work.

I had a question. When reshaping X for lstm (samples,timesteps,features) why did you model the problem

it be timesteps = lag window size

and the output dense layer have the size of horizon_window. This will give much better results in my opinion

Here is a link which will make my question more clear:

<https://stackoverflow.com/questions/42585356/how-to-construct-input-data-to-lstm-for-time-series-multi-step-horizon-with-exte>

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Jason Brownlee August 17, 2017 at 4:54 pm #

REPLY ↩

I model the problem with no timesteps and lots of features (multiple obs at the same time).

Get Your Start in Machine Learning

I found that if you frame the problem with multiple time steps for multiple features, performance was worse. Basically, we are using the LSTM as an MLP-type network here.

LSTMs are not great at autoregression, but this post was the most requested I've ever had.

More on LSTM suitability here:

<http://machinelearningmastery.com/suitability-long-short-term-memory-networks-time-series-forecasting/>



Jaskaran August 18, 2017 at 5:59 am #

REPLY ↩

So Jason,

Correct me if I am wrong but the whole point of RNN+LSTM learning over time(hidden states de
Essentially, this is just an autoregressive neural network. There is no storage of states over time



Jason Brownlee August 18, 2017 at 6:31 am #

Yes, there is no BPTT because we are only feeding in one time step.

You can add more history, but results will be worse. It turns out that LSTMs are poor at auto
<http://machinelearningmastery.com/suitability-long-short-term-memory-networks-time-series>

Nevertheless, I get a lot of people asking how to do it, so here it is.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**
Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



hanoun August 18, 2017 at 11:37 am #

REPLY ↩

Hi, I try to use this example to identify the shape switch an angle , its useful to use this tutorial and how I can test the model I train it,

Regards,
Hanan

Get Your Start in Machine Learning



A August 19, 2017 at 7:53 am #

REPLY ↩

Hi there – I love your blog and these tutorials! They're really helpful.

I have been studying both this tutorial and this one: <http://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/>. I have applied both codes to a simple dataset I'm working with (date, ROI%). Both codes run fine with my data, but I'm having a problem that has me completely stumped:

With this code, I'm able to actually forecast the future ROI%. With the other, it does a lot better at modeling the past data, but I can't figure out how to get it to forecast the future. Both codes have elements I need, but I can't seem to figure out how to bring them together.

Any insight would be awesome! Thank you!



Jason Brownlee August 20, 2017 at 6:04 am #

What is the problem exactly?



Ankit August 22, 2017 at 11:34 pm #

Jason, first of all, I would like to thank you for the work you've done. It has been tremendously helpful.

I have a question and seeking your expert opinion.

How to handle a time series data set with multiple and variable granularity input of each time step. for instance

Date | Area | Product category | Orders | Revenue | Cost

so, in this case, there would be multiple records for a single day aggregated on date and this is the granularity I want.

How should this kind of data be handled, since these features will contribute to the Revenue and Orders?

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**. Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Jason Brownlee August 23, 2017 at 6:53 am #

Get Your Start in Machine Learning



You could standardize the data and feed it into one model or build separate models and combine their predictions.

Try a few methods and see what works best for your problem.



Daniel August 24, 2017 at 2:07 am #

REPLY ↩

I am using this framework for my first shot at an LSTM network for monitoring network response times. The data I'm working with currently is randomly generated by simulating API calls. What I'm seeing is the LSTM seems to always predict a return to what looks like the mean of the data. Is this a function of the data being stochastic?

Separate question: since LSTM's have a memory component built into the neurons, what are the advantages/disadvantages of using a larger n_{in}/n_{lag} than 1?



Jason Brownlee August 24, 2017 at 6:48 am #

The problem might be too hard for your model, perhaps tune the LSTM or try another algo

A key benefit of LSTMs is that they the lag can extend much longer than other methods, e.g. hundre something like:

$\hat{y} = f(t-1, \dots, t-500)$

And the model can reproduce something it saw 500 time steps ago if needed.

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Daniel August 26, 2017 at 3:34 am #

REPLY ↩

Thanks. I am playing with some toy data now just to make sure I'm understanding how this works.

I am able to model a cosine wave very nicely with a 5 neuron, 100 epoch training run against `np.cos(range(100))` split into 80/20 training set. This is with the scaling, but without the difference. I feed in 10 inputs, and get 30 outputs.

Get Your Start in Machine Learning

Does calling `model.predict` change the model? I am calling repeatedly with the same 10 inputs and am seeing a different result each time. It looks like the predicted wave cycles through different amplitudes.



Daniel August 26, 2017 at 4:09 am #

REPLY ↩

Ah ok, I got it. Since `stateful` is on, I would need to do an explicit `reset_states` between predictions. Makes sense, I think! `Stateful` was useful for training, but since I won't be "online learning" and since I feed the network lag in the features, I should not rely on state for predictions.



Jason Brownlee August 26, 2017 at 6:48 am #

Nice work!

Yes, generally scaling is important, but if your cosine wave values are in $[0,1]$ then you're good.



Daniel August 26, 2017 at 6:03 am #

I have a simple question. Trying to set up an a different toy problem, with data generated (no validation). No matter how many layers, neurons, epochs that I train over, the results tend to be lower values, but it diverges quickly and approaches some fixed $y=400$ for higher values.

Do you have any ideas why this would happen?



Jason Brownlee August 26, 2017 at 6:51 am #

REPLY ↩

Maybe error accumulating. You're giving the LSTM a hard time.

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning



Daniel September 1, 2017 at 2:47 am #

REPLY ↩

Can I get your input on this issue I'm having? I would really like to make sure that I'm not implementing incorrectly. If there are network parameters I need to do, I can go through that exercise. But, I am not feeling confident about what I am on the right path with this problem.

<https://stackoverflow.com/questions/45982445/keras-lstm-time-series-multi-step-predictions-has-same-output-for-any-input>



Lucius September 1, 2017 at 6:14 pm #

REPLY ↩

Hi, there is a problem with the code. when doing data processing, i.e. calculate difference and min max scale. you should not use all data. in more real situation, you can only do this to train data. since you have no idea about test data.

So I changed the code, cut the last 12 month as test. then only use 24 months data for difference, min max scale, fit the model and predict for month 27.

Then I continue to use 25 months data for difference, min max scale, fit the model and predict for month 27. ...

The final result is worse than baseline.!

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Jason Brownlee September 2, 2017 at 6:04 am #

Correct, this is a simplification I implemented to keep the tutorial short and understandable.



Eldar M. September 17, 2017 at 1:47 am #

REPLY ↩

Hi Jason, I was able to get slightly better results with a custom loss function (weighted mse)

```
def weighted_mse(yTrue,yPred):
```

Get Your Start in Machine Learning

```
ones = K.ones_like(yTrue[0,:])
idx = K.cumsum(ones)
return K.mean((1/idx)*K.square(yTrue-yPred))
```

credit goes to Daniel Möller on Stack Overflow as I was not able to figure out the tensor modification steps on my own and he responded to my question there



Jason Brownlee September 17, 2017 at 5:28 am #

REPLY ↩

Nice one! Thanks for sharing.



Alex September 23, 2017 at 1:53 am #

```
def make_forecasts(model, n_batch, train, test, n_lag, n_seq):
    forecasts = list()
    for i in range(len(test)):
        X, y = test[i, 0:n_lag], test[i, n_lag:]
        # make forecast
        forecast = forecast_lstm(model, X, n_batch)
        # store the forecast
        forecasts.append(forecast)
    return forecasts
```

What is the point of the “train” data set as parameter in this function if it is not used?

Thanks

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Jason Brownlee September 23, 2017 at 5:43 am #

REPLY ↩

Yep, looks like its not used. You can probably remove it.

Get Your Start in Machine Learning



Fei September 24, 2017 at 1:51 am #

REPLY ↩

Hello, It is very useful tutorial. I am starter for the python and programming. May I convert input of model into 4 or more than one variable? and change the n_batch into other number not 1?



Jason Brownlee September 24, 2017 at 5:17 am #

REPLY ↩

Sure.



Fei September 26, 2017 at 4:33 am #

But ,When I change the n_batch size, the model does not work. By the way, you said manually



Fabian September 29, 2017 at 7:41 pm #

Hi Jason,
thanks a lot for your tutorials on LSTMs.
Do you have a suggestion how to model the network for a multivariate multi-step forecast? I read your a combining both seems to be more tricky as the output of the dense layer gets a higher dimension.

In words of your example here: if I want to forecast not only shampoo but also toothpaste sales T time steps ahead, now can I achieve the forecast to have the dimension 2xT? Is there an alternative to the dense layer?

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Jason Brownlee September 30, 2017 at 7:38 am #

REPLY ↩

I see. You could have two neurons in the output layer of your network, as easy as that.

Get Your Start in Machine Learning



Camille September 30, 2017 at 9:07 am #

REPLY ↩

Thanks for this great tutorial. Do you think this technique is applicable on the case of a many-to-many prediction?

A toy scenario: Imagine a machine with has 5 tuning knobs $[x_1, x_2, x_3, x_4, x_5]$ and as a result we can read 2 values $[y, z]$ as a response to a change of any of the knobs.

I am wondering if I can use LSTM to predict y and z at with a single model instead of building one model for y and another for z ? I am planning to follow this tutorial but I will love to hear what you think about it.



Jason Brownlee October 1, 2017 at 9:02 am #

Yes, LSTMs can easily be configured to support multiple input series and output a vector or

For example of taking multiple series as input, see this post:

<https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/>

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Jean-Marc September 30, 2017 at 12:08 pm #

Hi Jason, thank you very much for this tutorial. I am just starting with LSTM and your series on

A question about multi-output forecasting: how to deal with a multi-output when plotting the true data ver

Let's say I have a model to forecast the next 10 steps ($t, t+1, \dots, t+9$).

Using the observation at time:

→ $t=0$, the model will give a forecast for $t=1, 2, 3, 4, 5, 6, 7, 8, 9, 10$

and similarly, at

→ $t=1$, a forecast will be output for $t=2, 3, 4, 5, 6, 7, 8, 9, 10, 11$

etc...

There is overlap in the timestep for the forecast from $t=0$ and from $t=1$. For example, if I want to know the value at $t=2$, should I use the forecast from $t=1$ or from $t=0$, or a weighted average of the forecast?

May be using only the forecast from $t=1$ enough, because it already includes the history of the time series

Get Your Start in Machine Learning



Jason Brownlee October 1, 2017 at 9:06 am #

REPLY ↩

I'm not sure I follow. Perhaps you might be better off starting with linear models then move to an LSTM to lift skill on a framing/problem that is already working:

<https://machinelearningmastery.com/start-here/#timeseries>



mr October 1, 2017 at 9:53 pm #

REPLY ↩

The:

```
return datetime.strptime('190'+x, '%Y-%m')
```

gives me:

ValueError: time data '1901/1' does not match format '%Y-%m'

Thanks in advance

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Jason Brownlee October 2, 2017 at 9:38 am #

Perhaps confirm that you downloaded the dataset in CSV format.



wmbm October 4, 2017 at 10:29 pm #

REPLY ↩

So you don't actually need to split the data into test and training sets because you don't use the training set in this code. So this then becomes an unsupervised problem?

Get Your Start in Machine Learning



Jason Brownlee October 5, 2017 at 5:23 am #

REPLY ↩

No, it is a supervised learning model.

We use walk-forward validation. Learn more about it here:

<https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/>



wmbm October 5, 2017 at 6:26 am #

REPLY ↩

my mistake, I was look at just the multi-step persistence model. Thanks!



Jason Brownlee October 5, 2017 at 5:15 pm #

No problem.

Leave a Reply

Name (required)

Get Your Start in Machine Learning



You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning

Email (will not be published) (required)

Website

[SUBMIT COMMENT](#)

Welcome to Machine Learning Mastery



Hi, I'm Dr. Jason Brownlee.

My goal is to make practitioners like YOU awesome at applied machine learning.

[Read More](#)

Deep Learning for Sequence Prediction

Cut through the math and research papers.
Discover 4 Models, 6 Architectures, and 14 Tutorials.

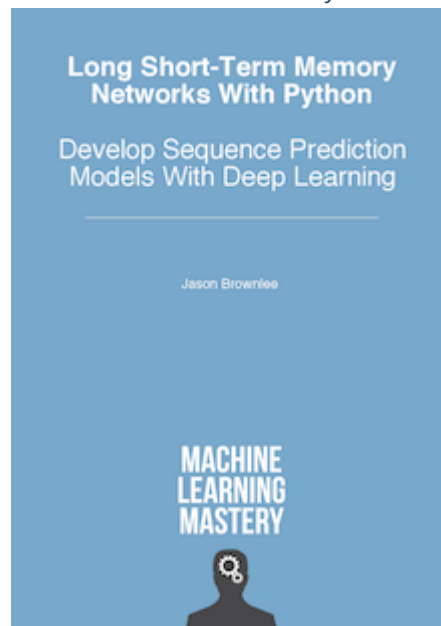
Get Your Start in Machine Learning



You can master applied Machine Learning
without the math or fancy degree.

Find out how in this *free* and *practical* email
course.

[START MY EMAIL COURSE](#)[Get Your Start in Machine Learning](#)

[Get Started With LSTMs in Python Today!](#)

POPULAR

**Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras**

JULY 21, 2016

**Your First Machine Learning Project in Python Step-By-Step**

JUNE 10, 2016

**Develop Your First Neural Network in Python With Keras Step-By-Step**

MAY 24, 2016

**Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras**

JULY 26, 2016

How to Setup a Python Environment for Machine Learning and Deep Learning with Anaconda

MARCH 13, 2017

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)[Get Your Start in Machine Learning](#)



Multi-Class Classification Tutorial with the Keras Deep Learning Library

JUNE 2, 2016



Time Series Forecasting with the Long Short-Term Memory Network in Python

APRIL 7, 2017



Regression Tutorial with the Keras Deep Learning Library in Python

JUNE 9, 2016



Multivariate Time Series Forecasting with LSTMs in Keras

AUGUST 14, 2017



How to Implement the Backpropagation Algorithm From Scratch In Python

NOVEMBER 7, 2016

Get Your Start in Machine Learning ×

You can master applied Machine Learning **without the math or fancy degree.** Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

© 2017 Machine Learning Mastery. All Rights Reserved.

[Privacy](#) | [Contact](#) | [About](#)

Get Your Start in Machine Learning