

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

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

# Stateful and Stateless LSTM for Time Series Forecasting with Python

by **Jason Brownlee** on April 21, 2017 in **Long Short-Term Memory Networks**



The Keras Python deep learning library supports both stateful and stateless Long Short-Term Memory (LSTM) networks.

When using stateful LSTM networks, we have fine-grained control over when the internal state of the LSTM network is reset. Therefore, it is important to understand different ways of managing this internal state when fitting and making predictions with LSTM networks affect the skill of the network.

In this tutorial, you will explore the performance of stateful and stateless LSTM networks in Keras for time series forecasting.

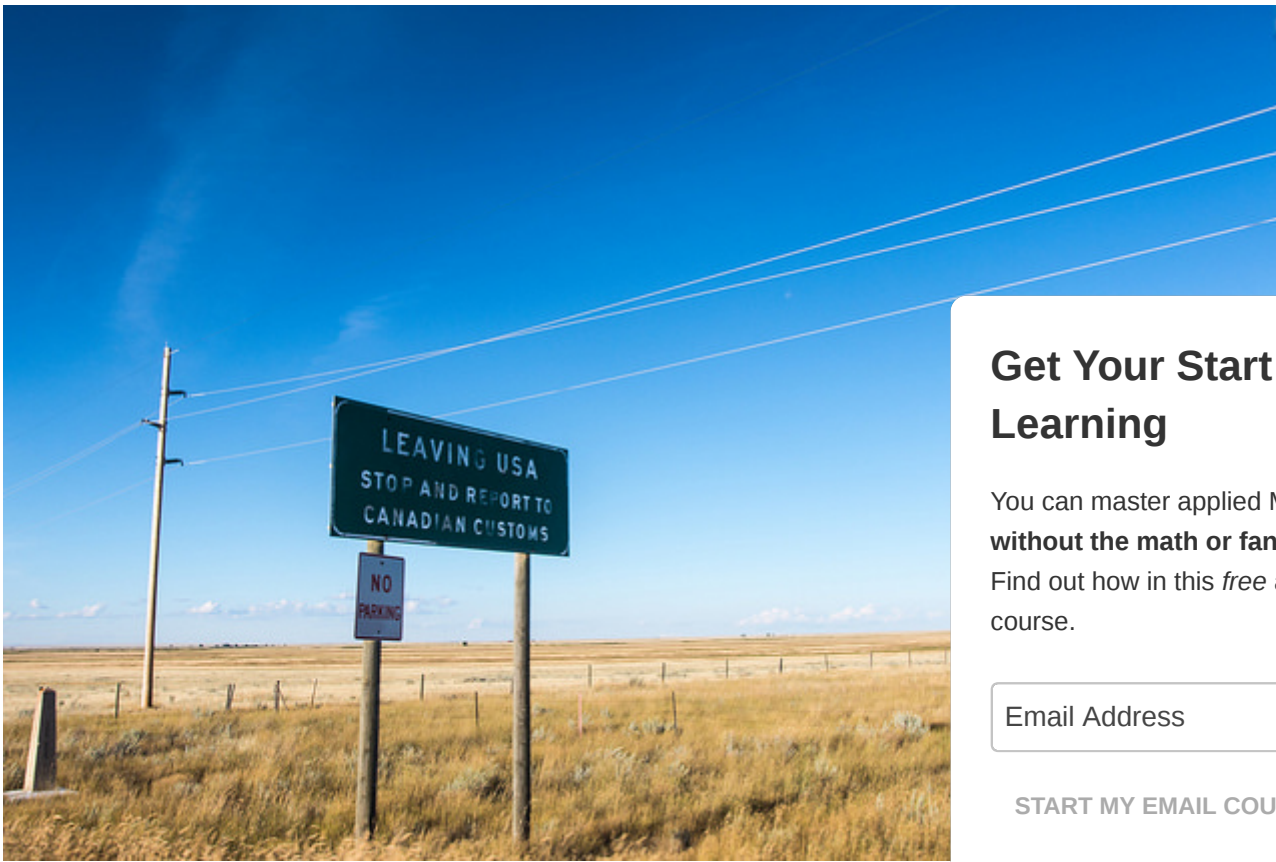
After completing this tutorial, you will know:

- How to compare and contrast stateful and stateless LSTM networks for time series forecasts.

[Get Your Start in Machine Learning](#)

- How the batch size in stateless LSTMs relate to stateful LSTM networks.
- How to evaluate and compare different state resetting regimes for stateful LSTM networks.

Let's get started.



Stateful and Stateless LSTM for Time Series Forecasting with Python  
Photo by [m01229](#), 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

This tutorial is broken down into 7 parts. They are:

1. Shampoo Sales Dataset

[Get Your Start in Machine Learning](#)

2. Experimental Test Harness
3. A vs A Test
4. Stateful vs Stateless
5. Stateless With Large Batch vs Stateless
6. Stateful Resetting vs Stateless
7. Review of Findings

## 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](#)

## 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

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, date_parser=parser)
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.

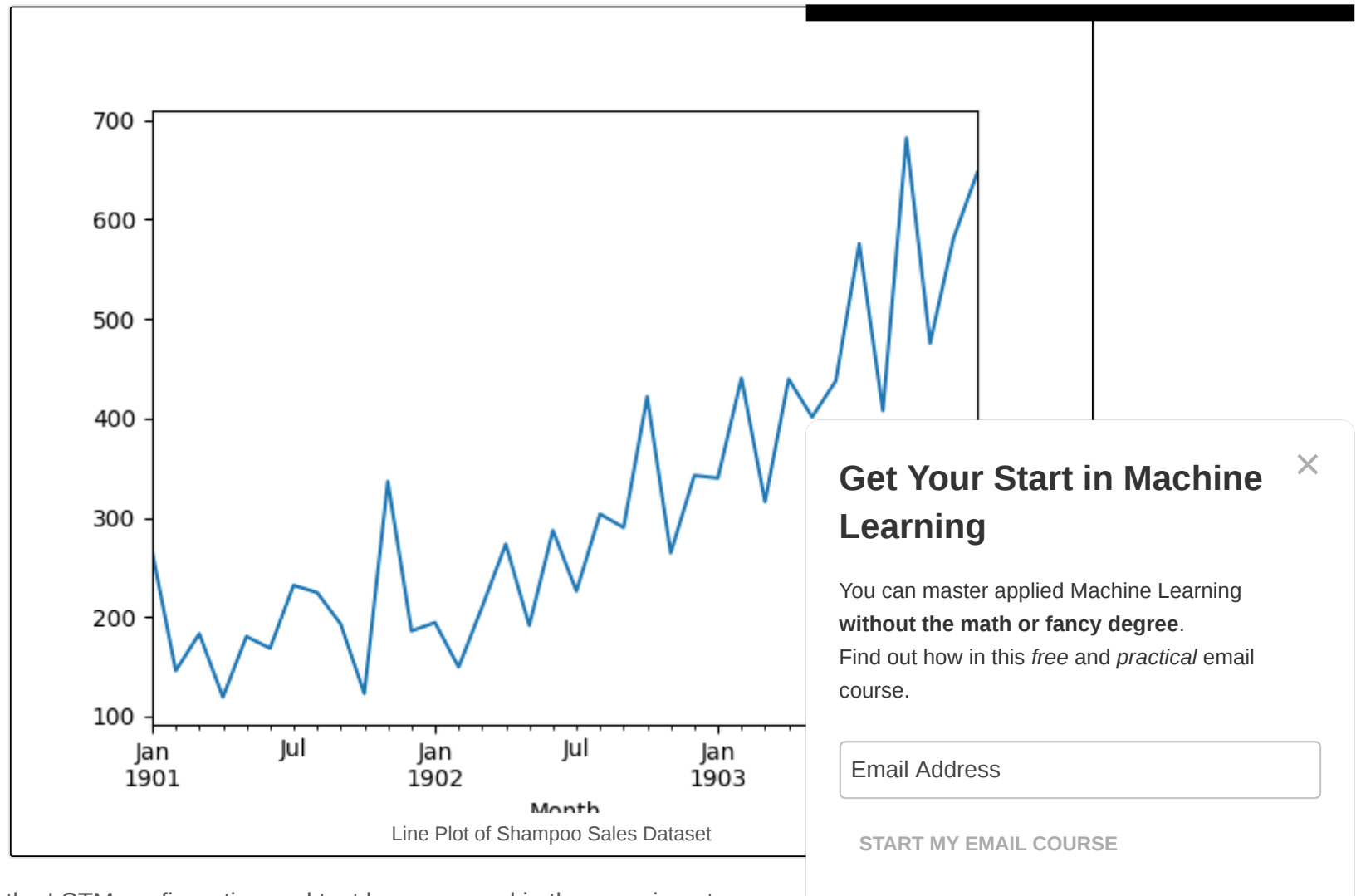
```
1 Month
2 1901-01-01 266.0
3 1901-02-01 145.9
4 1901-03-01 183.1
5 1901-04-01 119.3
6 1901-05-01 180.3
7 Name: Sales, dtype: float64
```

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



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

## Experimental Test Harness

This section describes the test harness 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.

The persistence forecast (naive forecast) on the test dataset achieves an error of 136.761 monthly shampoo sales. This provides a lower acceptable bound of performance on the test set.

## 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 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.

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 will be used as it punishes large errors and results in a score that is in the same units as the forecast.

## Data Preparation

Before we can fit an LSTM model to the dataset, we must transform the data.

The following three data transforms are performed on the dataset prior to fitting a model and making a forecast.

1. **Transform the time series data so that it is stationary.** Specifically, a lag=1 differencing to remove the increasing trend in the data.
2. **Transform the time series into a supervised learning problem.** Specifically, the organization of data into input and output patterns where the observation at the previous time step is used as an input to forecast the observation at the current time step

### 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

3. **Transform the observations to have a specific scale.** Specifically, to rescale the data to values between -1 and 1 to meet the default hyperbolic tangent activation function of the LSTM model.

These transforms are inverted on forecasts to return them into their original scale before calculating an error score.

## LSTM Model

We will use a base stateful LSTM model with 1 neuron fit for 1000 epochs.

A batch size of 1 is required as we will be using walk-forward validation and making one-step forecasts for each of the final 12 months of test data.

A batch size of 1 means that the model will be fit using online training (as opposed to batch training or mini-batch training). As a result, it is expected that the model fit will have some variance.

Ideally, more training epochs would be used (such as 1500), but this was truncated to 1000 to keep the training time reasonable.

The model will be fit using the efficient ADAM optimization algorithm and the mean squared error loss.

## Experimental Runs

Each experimental scenario will be run 10 times.

The reason for this is that the random initial conditions for an LSTM network can result in very different results.

Let's dive into the experiments.

## A vs A Test

A good first experiment is to evaluate how noisy or reliable our test harness may be.

This can be evaluated by running the same experiment twice and comparing the results. This is often called an A vs A test in the world of [A/B testing](#), and I find this name useful. The idea is to flush out any obvious faults with the experiment and get a handle on the expected variance in the mean value.

We will run an experiment with a stateful LSTM on the network twice.

### 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

The complete code listing is provided below.

This code also provides the basis for all experiments in this tutorial. Rather than re-listing it for each variation in subsequent sections, I will only list the functions that have been changed.

```
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
9 from keras.layers import Dense
10 from keras.layers import LSTM
11 from math import sqrt
12 import matplotlib
13 import numpy
14 from numpy import concatenate
15
16 # date-time parsing function for loading the dataset
17 def parser(x):
18     return datetime.strptime('190'+x, '%Y-%m')
19
20 # frame a sequence as a supervised learning problem
21 def timeseries_to_supervised(data, lag=1):
22     df = DataFrame(data)
23     columns = [df.shift(i) for i in range(1, lag+1)]
24     columns.append(df)
25     df = concat(columns, axis=1)
26     return df
27
28 # create a differenced series
29 def difference(dataset, interval=1):
30     diff = list()
31     for i in range(interval, len(dataset)):
32         value = dataset[i] - dataset[i - interval]
33         diff.append(value)
34     return Series(diff)
35
36 # invert differenced value
37 def inverse_difference(history, yhat, interval=1):
38     return yhat + history[-interval]
39
40 # scale train and test data to [-1, 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



```
41 def scale(train, test):
42     # fit scaler
43     scaler = MinMaxScaler(feature_range=(-1, 1))
44     scaler = scaler.fit(train)
45     # transform train
46     train = train.reshape(train.shape[0], train.shape[1])
47     train_scaled = scaler.transform(train)
48     # transform test
49     test = test.reshape(test.shape[0], test.shape[1])
50     test_scaled = scaler.transform(test)
51     return scaler, train_scaled, test_scaled
52
53 # inverse scaling for a forecasted value
54 def invert_scale(scaler, X, yhat):
55     new_row = [x for x in X] + [yhat]
56     array = numpy.array(new_row)
57     array = array.reshape(1, len(array))
58     inverted = scaler.inverse_transform(array)
59     return inverted[0, -1]
60
61 # fit an LSTM network to training data
62 def fit_lstm(train, batch_size, nb_epoch, neurons):
63     X, y = train[:, 0:-1], train[:, -1]
64     X = X.reshape(X.shape[0], 1, X.shape[1])
65     model = Sequential()
66     model.add(LSTM(neurons, batch_input_shape=(batch_size, X.shape[1], X.shape[2]), stateful=True))
67     model.add(Dense(1))
68     model.compile(loss='mean_squared_error', optimizer='adam')
69     for i in range(nb_epoch):
70         model.fit(X, y, epochs=1, batch_size=batch_size, verbose=0, shuffle=False)
71         model.reset_states()
72     return model
73
74 # make a one-step forecast
75 def forecast_lstm(model, batch_size, X):
76     X = X.reshape(1, 1, len(X))
77     yhat = model.predict(X, batch_size=batch_size)
78     return yhat[0,0]
79
80 # run a repeated experiment
81 def experiment(repeats, series):
82     # transform data to be stationary
83     raw_values = series.values
84     diff_values = difference(raw_values, 1)
85     # transform data to be supervised learning
86     supervised = timeseries_to_supervised(diff_values, 1)
87     supervised_values = supervised.values[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

```

88 # split data into train and test-sets
89 train, test = supervised_values[0:-12, :], supervised_values[-12:, :]
90 # transform the scale of the data
91 scaler, train_scaled, test_scaled = scale(train, test)
92 # run experiment
93 error_scores = list()
94 for r in range(repeats):
95     # fit the base model
96     lstm_model = fit_lstm(train_scaled, 1, 1000, 1)
97     # forecast test dataset
98     predictions = list()
99     for i in range(len(test_scaled)):
100         # predict
101         X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
102         yhat = forecast_lstm(lstm_model, 1, X)
103         # invert scaling
104         yhat = invert_scale(scaler, X, yhat)
105         # invert differencing
106         yhat = inverse_difference(raw_values, yhat, len(test_scaled)+1-i)
107         # store forecast
108         predictions.append(yhat)
109     # report performance
110     rmse = sqrt(mean_squared_error(raw_values[-12:], predictions))
111     print('%d) Test RMSE: %.3f' % (r+1, rmse))
112     error_scores.append(rmse)
113 return error_scores
114
115 # execute the experiment
116 def run():
117     # load dataset
118     series = read_csv('shampoo-sales.csv', header=0, parse_dates=[0], index_col=0, squeeze=True)
119     # experiment
120     repeats = 10
121     results = DataFrame()
122     # run experiment
123     results['results'] = experiment(repeats, series)
124     # summarize results
125     print(results.describe())
126     # save results
127     results.to_csv('experiment_stateful.csv', index=False)
128
129 # entry point
130 run()

```

## 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

Running the experiment saves the results to a file named “*experiment\_stateful.csv*”.

Get Your Start in Machine Learning

Run the experiment a second time and change the filename written by the experiment to “*experiment\_stateful2.csv*” as to not overwrite the results from the first run.

You should now have two sets of results in the current working directory in the files:

- *experiment\_stateful.csv*
- *experiment\_stateful2.csv*

We can now load and compare these two files. The script to do this is listed below.

```
1 from pandas import DataFrame
2 from pandas import read_csv
3 from matplotlib import pyplot
4 # load results into a dataframe
5 filenames = ['experiment_stateful.csv', 'experiment_stateful2.csv']
6 results = DataFrame()
7 for name in filenames:
8     results[name[11:-4]] = read_csv(name, header=0)
9 # describe all results
10 print(results.describe())
11 # box and whisker plot
12 results.boxplot()
13 pyplot.show()
```

This script loads the result files and first calculates descriptive statistics for each run.

We can see that the mean results and standard deviation are relatively close values (around 103-106) perfect. It is expected that increasing the number of repeats of the experiment from 10 to 30, 100, or more will give more accurate statistics.

1		stateful	stateful2
2	count	10.000000	10.000000
3	mean	103.142903	106.594624
4	std	7.109461	10.687509
5	min	94.052380	91.570179
6	25%	96.765985	101.015403
7	50%	104.376252	102.425406
8	75%	107.753516	115.024920
9	max	114.958430	125.088436

The comparison also creates a box and whisker plot to compare the two distributions.

## Get Your Start in Machine Learning

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

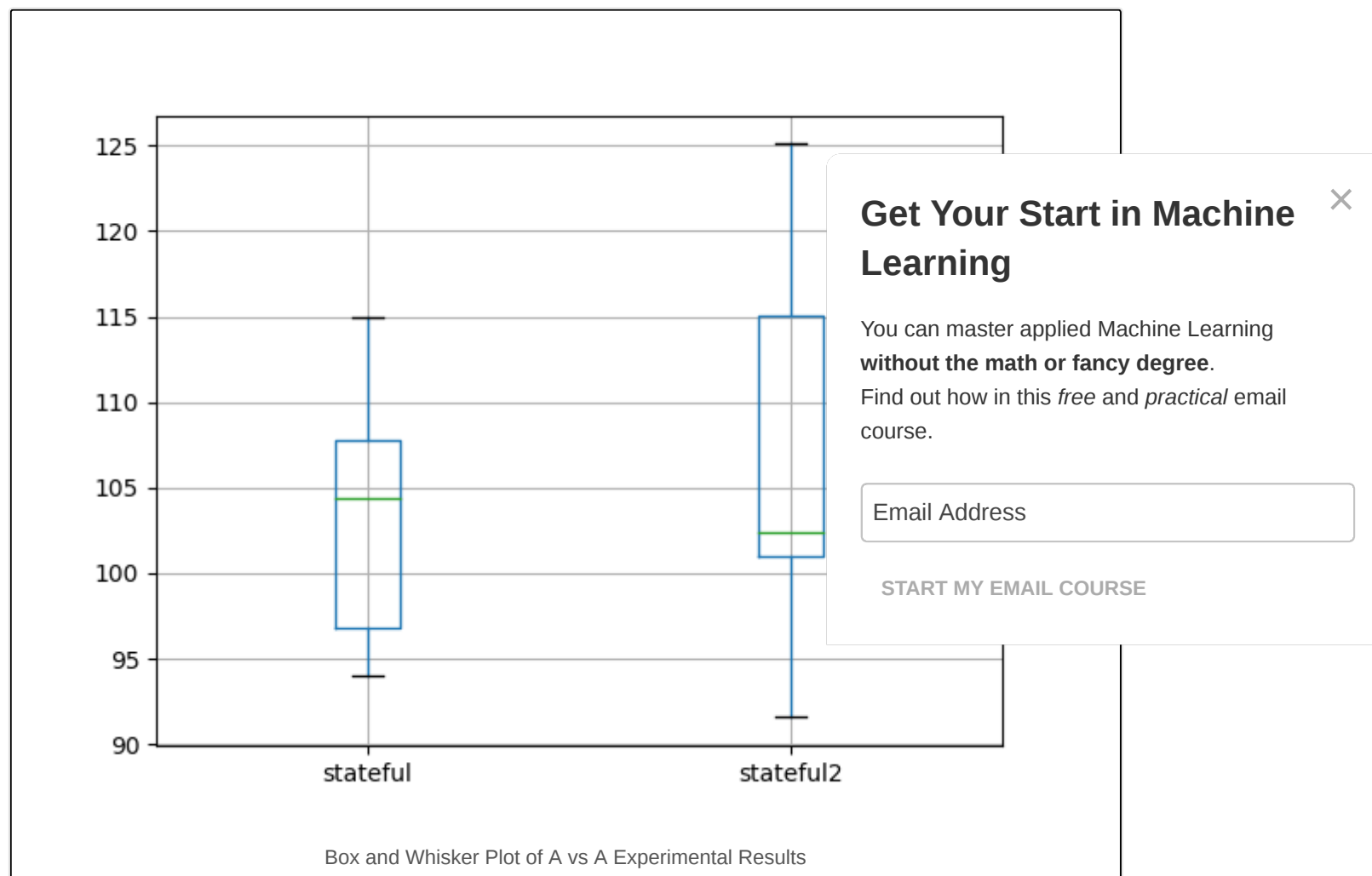
Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

The plot shows the 25th, 50th (median), and 75th percentile of 10 test RMSE results from each experiment. The box shows the middle 50% of the data and the green line shows the median.

The plot shows that although the descriptive statistics are reasonably close, the distributions do show some differences.

Nevertheless, the distributions do overlap and comparing means and standard deviations of different experimental setups is reasonable as long as we don't quibble over modest differences in mean.



## 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

A good follow-up to this analysis is to review the standard error of the distribution with different sample sizes. This would involve first creating a larger pool of experimental runs from which to draw (100 or 1000), and would give a good idea of a robust number of repeats and an expected error on the mean when comparing results.

## Stateful vs Stateless LSTMs

A good first experiment is to explore whether maintaining state in the LSTM adds value over not maintaining state.

In this section, we will contrast:

1. A Stateful LSTM (first result from the previous section).
2. A Stateless LSTM with the same configuration.
3. A Stateless LSTM with shuffling during training.

The benefit of LSTM networks is their ability to maintain state and learn a sequence.

- **Expectation 1:** The expectation is that the stateful LSTM will outperform the stateless LSTM.

Shuffling of input patterns each batch or epoch is often performed to improve the generalizability of a model. A stateless LSTM does not shuffle input patterns during training because the network aims to learn the sequence of patterns without shuffling.

- **Expectation 2:** The expectation is that the stateless LSTM without shuffling will outperform the stateful LSTM.

The code changes to the stateful LSTM example above to make it stateless involve setting *stateless* to *True* and *shuffle* to *True* during training epoch training rather than manual. The results are written to a new file named “*experiment\_stateless*” below.

```
1 # fit an LSTM network to training data
2 def fit_lstm(train, batch_size, nb_epoch, neurons):
3     X, y = train[:, 0:-1], train[:, -1]
4     X = X.reshape(X.shape[0], 1, X.shape[1])
5     model = Sequential()
6     model.add(LSTM(neurons, batch_input_shape=(batch_size, X.shape[1], X.shape[2]), stateful=False))
7     model.add(Dense(1))
8     model.compile(loss='mean_squared_error', optimizer='adam')
9     model.fit(X, y, epochs=nb_epoch, batch_size=batch_size, verbose=0, shuffle=False)
```

### 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

```
10 return model
```

The stateless with shuffling experiment involves setting the *shuffle* argument to *True* when calling fit in the *fit\_lstm()* function. The results from this experiment are written to the file “*experiment\_stateless\_shuffle.csv*”.

The complete updated *fit\_lstm()* function is listed below.

```
1 # fit an LSTM network to training data
2 def fit_lstm(train, batch_size, nb_epoch, neurons):
3     X, y = train[:, 0:-1], train[:, -1]
4     X = X.reshape(X.shape[0], 1, X.shape[1])
5     model = Sequential()
6     model.add(LSTM(neurons, batch_input_shape=(batch_size, X.shape[1], X.shape[2]), stateful=False))
7     model.add(Dense(1))
8     model.compile(loss='mean_squared_error', optimizer='adam')
9     model.fit(X, y, epochs=nb_epoch, batch_size=batch_size, verbose=0, shuffle=True)
10    return model
```

After running the experiments, you should have three result files for comparison:

- *experiment\_stateful.csv*
- *experiment\_stateless.csv*
- *experiment\_stateless\_shuffle.csv*

We can now load and compare these results. The complete example for comparing the results is list

```
1 from pandas import DataFrame
2 from pandas import read_csv
3 from matplotlib import pyplot
4 # load results into a dataframe
5 filenames = ['experiment_stateful.csv', 'experiment_stateless.csv',
6             'experiment_stateless_shuffle.csv']
7 results = DataFrame()
8 for name in filenames:
9     results[name[11:-4]] = read_csv(name, header=0)
10 # describe all results
11 print(results.describe())
12 # box and whisker plot
13 results.boxplot()
14 pyplot.show()
```

Running the example first calculates and prints descriptive statistics for each of the experiments.

## 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 average results suggest that the stateless LSTM configurations may outperform the stateful configuration. If robust, this finding is quite surprising as it does not meet the expectation of the addition of state improving performance.

The shuffling of training samples does not appear to make a large difference to the stateless LSTM. If the result is robust, the expectation of shuffled training order on the stateless LSTM does appear to offer some benefit.

Together, these findings may further suggest that the chosen LSTM configuration is focused more on learning input-output pairs rather than dependencies within the sequence.

From these limited results alone, one would consider exploring stateless LSTMs on this problem.

	stateful	stateless	stateless_shuffle
1 count	10.000000	10.000000	10.000000
2 mean	103.142903	95.661773	96.206332
3 std	7.109461	1.924133	2.138610
4 min	94.052380	94.097259	93.678941
5 25%	96.765985	94.290720	94.548002
6 50%	104.376252	95.098050	95.804411
7 75%	107.753516	96.092609	97.076086
8 max	114.958430	100.334725	99.870445

A box and whisker plot is also created to compare the distributions.

The spread of the data appears much larger with the stateful configuration compared to the stateless statistics when we look at the standard deviation scores.

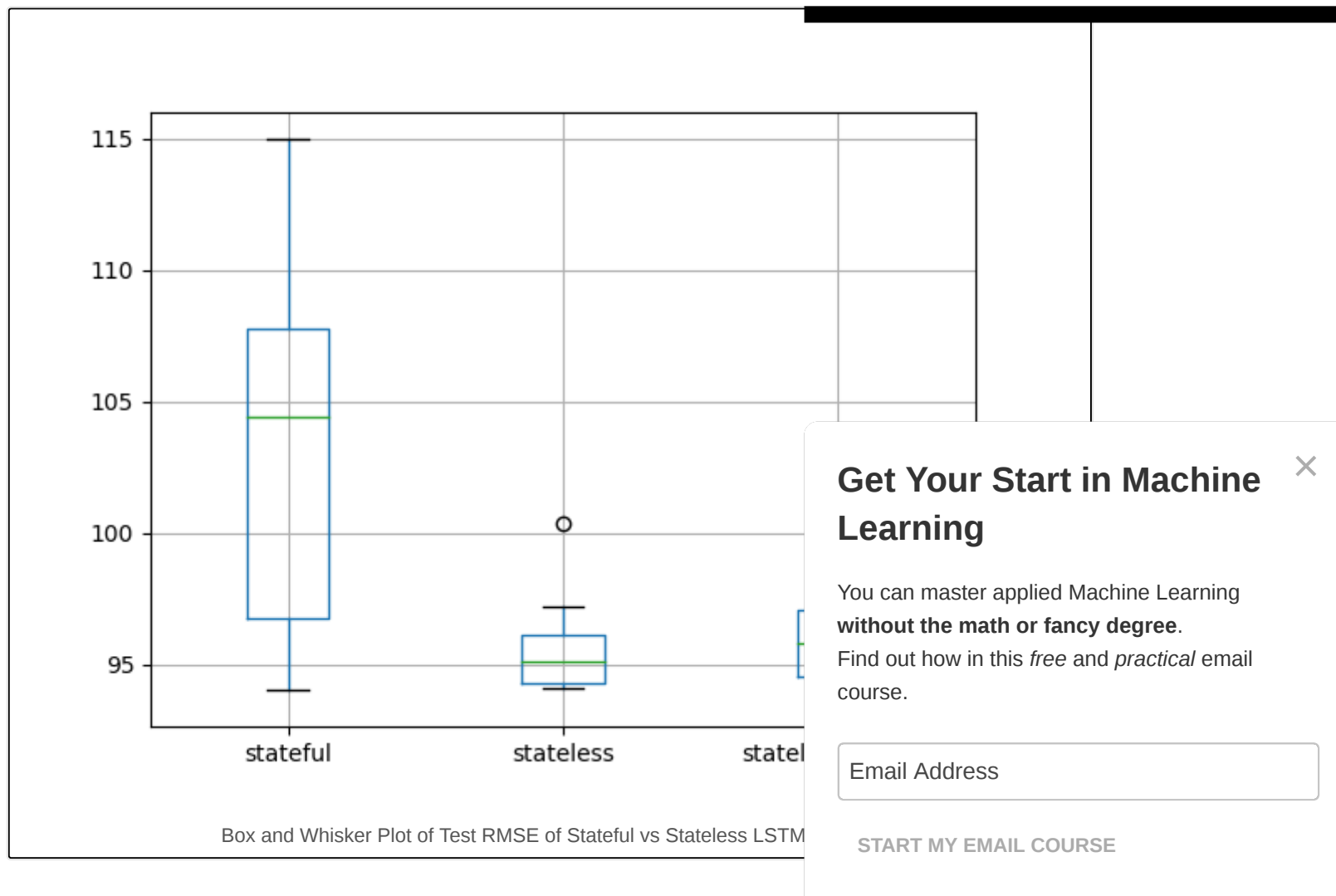
This suggests that the stateless configurations may be more stable.

## 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



## Stateless with Large Batch vs Stateless

A key to understanding the difference between stateful and stateless LSTMs is “when internal state is reset”.

- **Stateless:** In the stateless LSTM configuration, internal state is reset after each training batch or each batch when making predictions.
- **Stateful:** In the stateful LSTM configuration, internal state is only reset when the `reset_state()` function is called.

If this is the only difference, then it may be possible to simulate a stateful LSTM with a stateless LSTM

Get Your Start in Machine Learning



- **Expectation 3:** Stateless and stateful LSTMs should produce near identical results when using the same batch size.

We can do this with the Shampoo Sales dataset by truncating the training data to only 12 months and leaving the test data as 12 months. This would allow a stateless LSTM to use a batch size of 12. If training and testing were performed in a one-shot manner (one function call), then it is possible that internal state of the “stateless” LSTM would not be reset and both configurations would produce equivalent results.

We will use the stateful results from the first experiment as a starting point. The `forecast_lstm()` function is modified to forecast one year of observations in a single step. The `experiment()` function is modified to truncate the training dataset to 12 months of data, to use a batch size of 12, and to process the batched predictions returned from the `forecast_lstm()` function. These updated functions are listed below. Results are written to the file “`experiment_stateful_batch12.csv`”.

```

1 # make a one-step forecast
2 def forecast_lstm(model, batch_size, X):
3     X = X.reshape(1, 1, len(X))
4     yhat = model.predict(X, batch_size=batch_size)
5     return yhat[0,0]
6
7 # run a repeated experiment
8 def experiment(repeats, series):
9     # transform data to be stationary
10    raw_values = series.values
11    diff_values = difference(raw_values, 1)
12    # transform data to be supervised learning
13    supervised = timeseries_to_supervised(diff_values, 1)
14    supervised_values = supervised.values[1:,:]
15    # split data into train and test-sets
16    train, test = supervised_values[-24:-12, :], supervised_values[-12:,:]
17    # transform the scale of the data
18    scaler, train_scaled, test_scaled = scale(train, test)
19    # run experiment
20    error_scores = list()
21    for r in range(repeats):
22        # fit the base model
23        batch_size = 12
24        lstm_model = fit_lstm(train_scaled, batch_size, 1000, 1)
25        # forecast test dataset
26        test_reshaped = test_scaled[:,0:-1]
27        test_reshaped = test_reshaped.reshape(len(test_reshaped), 1, 1)
28        output = lstm_model.predict(test_reshaped, batch_size=batch_size)
29        predictions = list()
30        for i in range(len(output)):
31            yhat = output[i,0]
32            X = test_scaled[i, 0:-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

```

33         # invert scaling
34         yhat = invert_scale(scaler, X, yhat)
35         # invert differencing
36         yhat = inverse_difference(raw_values, yhat, len(test_scaled)+1-i)
37         # store forecast
38         predictions.append(yhat)
39     # report performance
40     rmse = sqrt(mean_squared_error(raw_values[-12:], predictions))
41     print('%d) Test RMSE: %.3f' % (r+1, rmse))
42     error_scores.append(rmse)
43     return error_scores

```

We will use the stateless LSTM configuration from the previous experiment with training pattern shuffling turned off as the starting point. The experiment uses the same *forecast\_lstm()* and *experiment()* functions listed above. Results are written to the file “*experiment\_stateless\_batch12.csv*”.

After running this experiment, you will have two result files:

- *experiment\_stateful\_batch12.csv*
- *experiment\_stateless\_batch12.csv*

We can now compare the results from these experiments.

```

1 from pandas import DataFrame
2 from pandas import read_csv
3 from matplotlib import pyplot
4 # load results into a dataframe
5 filenames = ['experiment_stateful_batch12.csv', 'experiment_stateless_batch12.csv']
6 results = DataFrame()
7 for name in filenames:
8     results[name[11:-4]] = read_csv(name, header=0)
9 # describe all results
10 print(results.describe())
11 # box and whisker plot
12 results.boxplot()
13 pyplot.show()

```

Running the comparison script first calculates and prints the descriptive statistics for each experiment.

The average results for each experiment suggest equivalent results between the stateless and stateful configurations with the same batch size. This confirms our expectations.

## 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

If this result is robust, it suggests that there are no further implementation-detailed differences between stateless and stateful LSTM networks in Keras beyond when the internal state is reset.

	stateful_batch12	stateless_batch12
1		
2 count	10.000000	10.000000
3 mean	97.920126	97.450757
4 std	6.526297	5.707647
5 min	92.723660	91.203493
6 25%	94.215807	93.888928
7 50%	95.770862	95.640314
8 75%	99.338368	98.540688
9 max	114.567780	110.014679

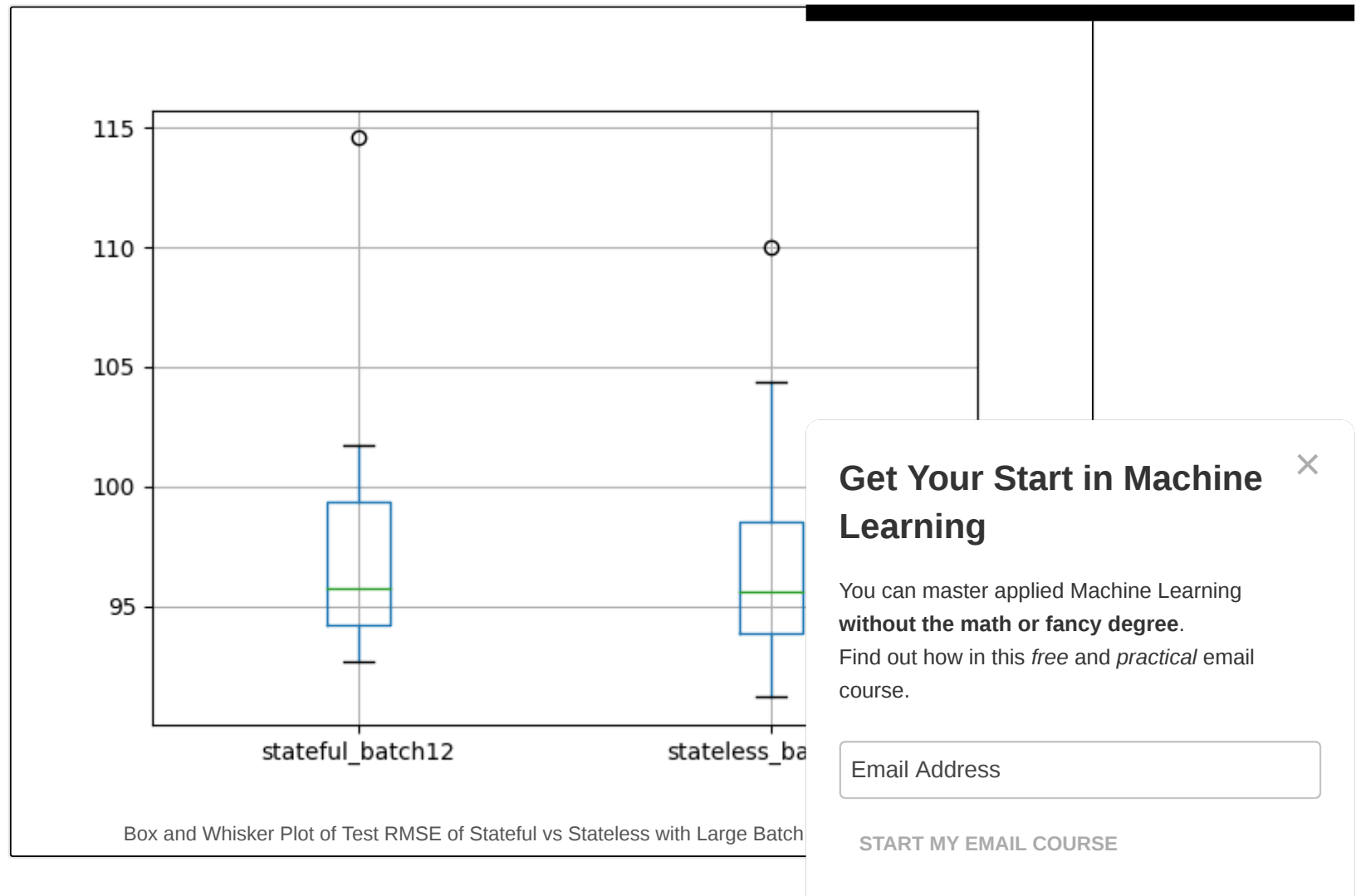
A box and whisker plot is also created to compare the distributions.

The plot confirms the story in the descriptive statistics, perhaps just highlighting variability in the exp

## 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



## Stateful Resetting vs Stateless

Another question regarding stateful LSTMs is the best regime to perform resets to state.

Generally, we would expect that resetting the state after each presentation of the sequence would be a good idea.

- **Expectation 4:** Resetting state after each training epoch results in better test performance.

Get Your Start in Machine Learning

This raises the question as to the best way to manage state when making predictions. For example, should the network be seeded with state from making predictions on the training dataset first?

- **Expectation 5:** Seeding state in the LSTM by making predictions on the training dataset results in better test performance.

We would also expect that not resetting LSTM state between one-step predictions on the test set would be a good idea.

- **Expectation 6:** Not resetting state between one-step predictions on the test set results in better test set performance.

There is also the question of whether or not resetting state at all is a good idea. In this section, we attempt to tease out answers to these questions.

We will again use all of the available data and a batch size of 1 for one-step forecasts.

In summary, we are going to compare the following experimental setups:

No Seeding:

- **noseed\_1:** Reset state after each training epoch and not during testing (the stateful results from
- **noseed\_2:** Reset state after each training epoch and after each one-step prediction (*experiment*
- **noseed\_3:** No resets after training or making one-step predictions (*experiment\_stateful\_norese*

Seeding:

- **seed\_1:** Reset state after each training epoch, seed state with one-step predictions on training and test dataset (*experiment\_stateful\_seed\_train.csv*).
- **seed\_2:** Reset state after each training epoch, seed state with one-step predictions on training and test dataset and reset state after each one-step prediction on train and test sets (*experiment\_stateful\_seed\_train\_and\_test.csv*).
- **seed\_3:** Seed on training dataset before making one-step predictions, no resets during training on predictions (*experiment\_stateful\_seed\_train\_no\_resets.csv*).

The stateful experiment code from the first “A vs A” experiment is used as a base.

The modifications needed for the various resetting/no-resetting and seeding/no-seeding are listed below.

## 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 update the `forecast_lstm()` function to update after each test by adding a call to `reset_states()` on the model after each prediction is made. The updated `forecast_lstm()` function is listed below.

```
1 # make a one-step forecast
2 def forecast_lstm(model, batch_size, X):
3     X = X.reshape(1, 1, len(X))
4     yhat = model.predict(X, batch_size=batch_size)
5     model.reset_states()
6     return yhat[0,0]
```

We can update the `fit_lstm()` function to not reset after each epoch by removing the call to `reset_states()`. The complete function is listed below.

```
1 # fit an LSTM network to training data
2 def fit_lstm(train, batch_size, nb_epoch, neurons):
3     X, y = train[:, 0:-1], train[:, -1]
4     X = X.reshape(X.shape[0], 1, X.shape[1])
5     model = Sequential()
6     model.add(LSTM(neurons, batch_input_shape=(batch_size, X.shape[1], X.shape[2]), stateful=True))
7     model.add(Dense(1))
8     model.compile(loss='mean_squared_error', optimizer='adam')
9     for i in range(nb_epoch):
10         model.fit(X, y, epochs=1, batch_size=batch_size, verbose=0, shuffle=False)
11     return model
```

We can seed the state of LSTM after training with the state from making predictions on the training data before making one-step forecasts. This can be added to the `run()` function before making one-step forecasts. The updated `run()` function is listed below.

```
1 # run a repeated experiment
2 def experiment(repeats, series):
3     # transform data to be stationary
4     raw_values = series.values
5     diff_values = difference(raw_values, 1)
6     # transform data to be supervised learning
7     supervised = timeseries_to_supervised(diff_values, 1)
8     supervised_values = supervised.values[1:,:]
9     # split data into train and test-sets
10    train, test = supervised_values[0:-12, :], supervised_values[-12:, :]
11    # transform the scale of the data
12    scaler, train_scaled, test_scaled = scale(train, test)
13    # run experiment
14    error_scores = list()
15    for r in range(repeats):
16        # fit the base model
```

## 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

```

17     lstm_model = fit_lstm(train_scaled, 1, 1000, 1)
18     # forecast train dataset
19     for i in range(len(train_scaled)):
20         X, y = train_scaled[i, 0:-1], train_scaled[i, -1]
21         yhat = forecast_lstm(lstm_model, 1, X)
22     # forecast test dataset
23     predictions = list()
24     for i in range(len(test_scaled)):
25         # predict
26         X, y = test_scaled[i, 0:-1], test_scaled[i, -1]
27         yhat = forecast_lstm(lstm_model, 1, X)
28         # invert scaling
29         yhat = invert_scale(scaler, X, yhat)
30         # invert differencing
31         yhat = inverse_difference(raw_values, yhat, len(test_scaled)+1-i)
32         # store forecast
33         predictions.append(yhat)
34     # report performance
35     rmse = sqrt(mean_squared_error(raw_values[-12:], predictions))
36     print('%d) Test RMSE: %.3f' % (r+1, rmse))
37     error_scores.append(rmse)
38     return error_scores

```

This concludes all of the piecewise modifications needed to create the code for these 6 experiments

After running these experiments you will have the following results files:

- *experiment\_stateful.csv*
- *experiment\_stateful\_reset\_test.csv*
- *experiment\_stateful\_noreset.csv*
- *experiment\_stateful\_seed\_train.csv*
- *experiment\_stateful\_seed\_train\_resets.csv*
- *experiment\_stateful\_seed\_train\_no\_resets.csv*

We can now compare the results, using the script below.

```

1 from pandas import DataFrame
2 from pandas import read_csv
3 from matplotlib import pyplot
4 # load results into a dataframe
5 filenames = ['experiment_stateful.csv', 'experiment_stateful_reset_test.csv',
6             'experiment_stateful_noreset.csv', 'experiment_stateful_seed_train.csv',
7             'experiment_stateful_seed_train_resets.csv', 'experiment_stateful_seed_train_no_resets.csv']

```

## 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 results = DataFrame()
9 for name in filenames:
10     results[name] = read_csv(name, header=0)
11 results.columns = ['noseed_1', 'noseed_2', 'noseed_3', 'seed_1', 'seed_2', 'seed_3']
12 # describe all results
13 print(results.describe())
14 # box and whisker plot
15 results.boxplot()
16 pyplot.show()

```

Running the comparison prints descriptive statistics for each set of results.

The results for no seeding suggest perhaps little difference between resetting after each prediction on the test dataset and not. This suggests any state built up from prediction to prediction is not adding value, or that this state is implicitly cleared by the Keras API. This was a surprising result.

The results on the no-seed case also suggest that having no resets during training results in worse performance than resetting the state at the end of each epoch. This confirms the expectation that resetting the state at the end of each epoch is better.

The average results from the seed experiments suggest that seeding LSTM state with predictions on the train and test datasets is neutral, if not resulting in slightly worse performance.

Resetting state after each prediction on the train and test sets seem to result in slightly better performance. Testing seems to result in the best performance.

These results regarding seeding are surprising, but we should note that the mean values are all within a few percent of each other, which could be statistical noise.

	noseed_1	noseed_2	noseed_3	seed_1	seed_2	seed_3
count	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000
mean	103.142903	101.757034	110.441021	105.468200	100.093551	98.766432
std	7.109461	14.584442	24.539690	5.206674	4.157095	11.573366
min	94.052380	91.264712	87.262549	97.683535	95.913385	90.005843
25%	96.765985	93.218929	94.610724	100.974693	96.721924	91.203879
50%	104.376252	96.144883	99.483971	106.036240	98.779770	95.079716
75%	107.753516	105.657586	121.586508	109.829793	103.082791	100.500867
max	114.958430	138.752321	166.527902	112.691046	108.070145	128.261354

A box and whisker plot is also created to compare the distributions.

## Get Your Start in Machine Learning

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

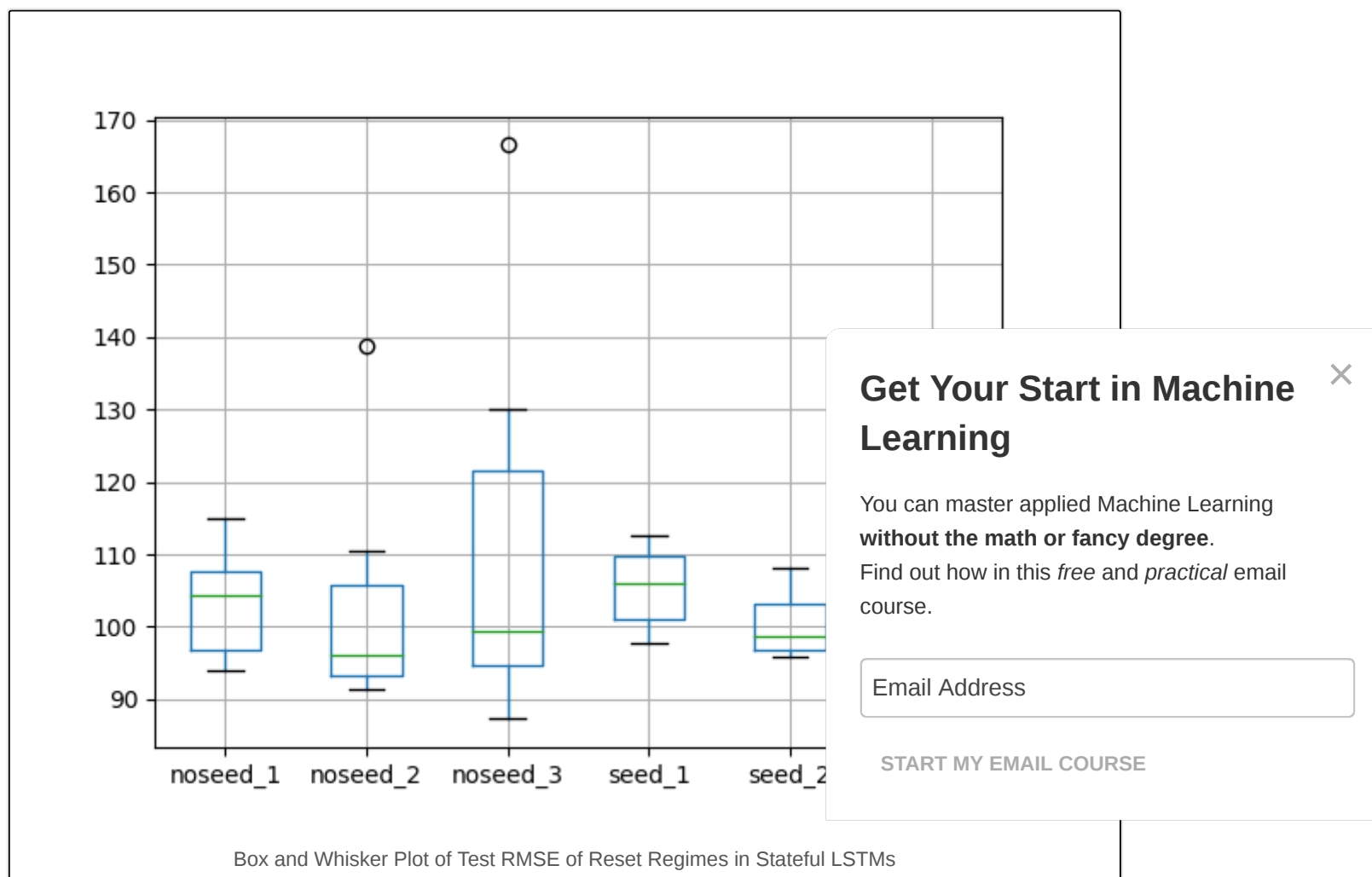
Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Get Your Start in Machine Learning



The plot tells the same story as the descriptive statistics. It highlights the increased spread when no resets are used on the stateless LSTM without seeding. It also highlights the general tight spread on the experiments that seed the state of the LSTM with predictions on the training dataset.



## Review of Findings

In this section, we recap the findings throughout this tutorial.

- 10 repeats of an experiment with the chosen configuration results in some variation in the mean and standard deviation of the test RMSE of about 3 monthly shampoo sales. More repeats would be expected to tighten this up.
- The stateless LSTM with the same configuration may perform better on this problem than the stateful version.
- Not shuffling training patterns with the stateless LSTM may result in slightly better performance.
- When a large batch size is used, a stateful LSTM can be simulated with a stateless LSTM.
- Resetting state when making one-step predictions with a stateful LSTM may improve performance on the test set.
- Seeding state in a stateful LSTM by making predictions on the training dataset before making predictions on the test set does not result in an obvious improvement in performance on the test set.
- Fitting a stateful LSTM and seeding it on the training dataset and not performing any resetting of state during training or prediction may result in better performance on the test set.

It must be noted that these findings should be made more robust by increasing the number of repeats of each experiment and confirming the differences are significant using statistical significance tests.

It should also be noted that these results apply to this specific problem, the way it was framed, and the topology, batch size, and training epochs.

## Summary

In this tutorial, you discovered how to investigate the impact of using stateful vs stateless LSTM networks.

Specifically, you learned:

- How to compare stateless vs stateful LSTM networks for time series forecasting.
- How to confirm the equivalence of stateless LSTMs and stateful LSTMs with a large batch size.
- How to evaluate the impact of when LSTM state is reset during training and making predictions with LSTM networks for time series forecasting.

Do you have any questions? Ask your questions in the comments 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!

Get Your Start in Machine Learning

## 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](#)

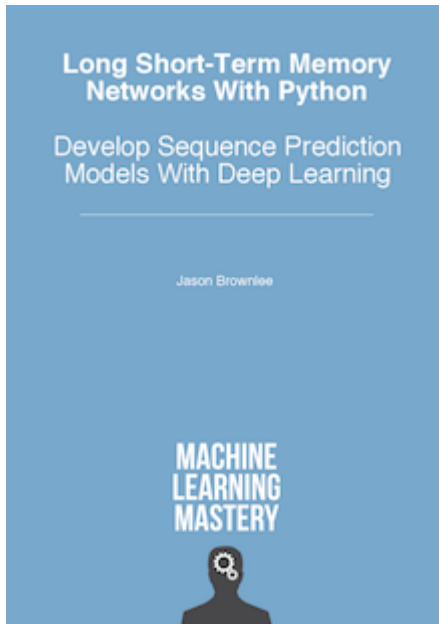
It provides **self-study tutorials** on topics like:

*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 Use Features in LSTM Networks for Time Series Forecasting](#)

[Instability of Online Learning for Stateful LSTM for Time Series Forecasting ▶](#)

[Get Your Start in Machine Learning](#)

## 16 Responses to *Stateful and Stateless LSTM for Time Series Forecasting with Python*



**Sam Taha** April 27, 2017 at 3:59 am #

REPLY ↩

Hi Jason,

Great subject and article.

How do we deal with the case when there the data has multiple features/labels per time series that we have suspicion have strong correlation (assumed to be strong). For example:

YR Month Production Sales

1999 1 Shampoo \$400

1999 1 Conditioner \$300

1999 2 Shampoo \$410

1999 2 Conditioner \$305

And so in this case we would like to be build one single model to predict Shampoo/Conditioner sales.

Is this configured by setting to the batch to 2 stateful (or 24 in the case of stateless) or am I looking at this

### 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** April 27, 2017 at 8:47 am #

Same as any other algorithm.

Explore using all features in the model, explore removing highly correlated features and see how that affects the model.



**chris** May 23, 2017 at 3:42 am #

REPLY ↩

Hi Jason,

I follow your blog entries regarding lstms and time series quite a while and I like them really well. I have

Get Your Start in Machine Learning

the blog posts, but maybe you would like to share your ideas with me as I am relatively new in this area. I have a record consisting of 61 multivariate timeseries. Each time I have assigned a label in the preprocessing (0, 1 or 2).

I would like to make a multi-class classification with lstm's.

As the starting point, I used the following:

```
model.add(LSTM(61, input_shape=(1, 61)))  
model.add(Dense(3, activation='softmax'))  
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

and learn with a batch size of 1 over 150 epochs. The dataset is divided into a training and test set.

I already have an accuracy of 97%. I think this is partly due to the data distribution, because label one occurs at about 94% and the other two about equally often. Nevertheless, it's better than ever predict label one.

In the next steps, I will first let the batch size vary, and manage the memory cell.

Do you have other suggestions to increase the accuracy or any other things to do differently based on your problem rather to a forecast problem?

Generally I have 3 problems:

- 1) The predicted labels are not 100% overlapping with the true labels. What I think is not quite so bad
- 2) I have something like jumping / oscillating values between two labels. I think that this will be improved a recommended standard technique to treat these labels again, as for example like the moving average
- 3) Completely wrongly labeled areas

Would you recommend to adjust the distribution of the data, so that each label equally often occur?

Many greetings,  
chris

## 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 23, 2017 at 7:58 am #

REPLY ↩

Hi Chris,

If your classes are not balanced, consider using a different measure than accuracy to get a true idea of the model skill, like a confusion matrix, logloss or AUC.

Try to quantify exactly what type of errors the model is making.

Get Your Start in Machine Learning

If there is conflict in the data, find a way to resolve it, perhaps the time axis or another variable can be used (feature engineering).

Most methods I know for rebalancing data are not for time series data, for example:

<http://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/>

I do have a suite of general ideas for lifting the skill of deep learning models here:

<http://machinelearningmastery.com/improve-deep-learning-performance/>

I hope that helps as a start.



**Chris** May 23, 2017 at 3:25 pm #

REPLY ↩

Good Morning, thanks for your quick reply. I started using the confusion matrix yesterday 😊. As for the feature engineering idea sounds really good, since I always have corridors of the same size for the C and first try the other approaches. Thanks for the advices! I will keep you up to date.



**Jason Brownlee** May 24, 2017 at 4:52 am #

Nice work, let me know how you go.



**chris** June 5, 2017 at 5:34 am #

Hi Jason,  
thanks again for the last tips.

Currently I get quite good results. The AUC for each of my three classes is already between 96-99 and also the F-Measure is better than in all models used so far.

I still have a few jumps in the labels, but I think I can certainly reduce them.

I now have the problem that I currently use 37 or even 62 different features.

I would like to perform a feature selection. Unfortunately, I have found nothing for keras / neural be used for neural networks or even has experience with it? Or is that no standard practice for N

## 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 5, 2017 at 7:43 am #

REPLY ↩

Hi chris,

I would recommend performing feature selection up-front (prior to modeling) using sklearn:  
<http://machinelearningmastery.com/feature-selection-machine-learning-python/>



**George Heitzer** May 30, 2017 at 8:53 am #

REPLY ↩

Hi Jason, when running the second ( large ) piece of code using PyCharm I get "UnicodeDecodeError: 'utf-8' codec can't decode byte 0x82 in position 22: invalid start byte", it seems to work using Spyder , however  
best regards George

## 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 2, 2017 at 12:31 pm #

I recommend running all code from the command line.



**Kris** June 22, 2017 at 12:11 am #

Hi Jason,

Great post! Maybe you would be able to help me with my problem?

I have a sequential data representing moving targets recorded by a radar. Sequences of some targets are longer than the others.

For example,

I have labeled data of cars and their velocities, accelerations etc.

Get Your Start in Machine Learning

'c' represents a car with a 3-dimensional feature vector, age is the number of times a target was recorded by the radar. Label means different types of cars such as truck etc.

c1 = [2,3,5], label = 0, age = 1

c2 = [2,4,7], label = 1, age = 1

c3 = [5,6,3], label = 2, age = 1

c1 = [4,5,7], label = 0, age = 2

c1 = [5,7,8], label = 0, age = 3

c2 = [6,7,4], label = 1, age = 2

c1 = [1,3,8], label = 0, age = 4

c3 = [5,6,3], label = 2, age = 2

As you can see, the sequences of some targets are longer than the others.

My question is how could I account for it while creating an LSTM model?

For example, by choosing window size 2, I would get [c1, c2, c3], [c2, c3, c1] and so on...

What happens to the labels in this case?

This is a classification problem so would stateful or stateless network be more appropriate?

Thank you,

Kris



**Jason Brownlee** June 22, 2017 at 6:08 am #

Sorry, I'm not sure I follow.

Consider providing the entire sequences as input time steps.

Also consider padding sequences to make the same length if the number of time steps differ.

## 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



**Tryfon** August 8, 2017 at 6:28 am #

Get Your Start in Machine Learning



Hi Jason! I have some second thoughts about the stateless lstm.

The main purpose of the LSTM is to utilize its memory property. Based on that what is the point of a stateless LSTM to exist? Don't we "convert" it into a simple NN by doing that?

In other words.. Does the stateless use of LSTM aim to model the sequences (window) in the input data – if we apply shuffle=False in the fit layer in keras – (eg. for a window of 10 time steps capture any pattern between 10-character words)? If yes why don't we convert the initial input data to match the form of the sequencers under inspection and then use a plain NN (by adding extra columns to the original dataset that are shifted)?

If we choose to have shuffle = True then we are losing any information that could be found in our data (e.g. time series data – sequences), don't we? In that case I would expect in to behave similarly to a plain NN and get the same results between the two by setting the same random seed.

Am I missing something in my thinking?



**Jason Brownlee** August 8, 2017 at 7:56 am #

The "stateless" LSTM just means that internal state is reset at the end of each batch, which

Really, maintaining state is part of the trade-off in backprop through time and input sequence length.

Shuffle applies to samples within a batch. BPTT really looks at time steps within a sample, then average

Does that help?



**Tryfon** August 8, 2017 at 10:41 pm #

Regarding the shuffling according to the documentation "shuffle: boolean or str (for 'batch'). Whether to shuffle the samples at each epoch". So it first resamples the data (i.e. changes the original order) and then on the new order creates the batches. Do I get it right?

Let me restate my previous question because I might have confused you. Suppose the dataset has only one variable X and one label Y. I actually want to know whether a stateless LSTM of batch size say 5 and timestep 1 is equivalent to a NN that will get as input X and X.shift(1) (so in total 2 inputs (2 columns) although they point to the same original X column of my dataset and batch size also 5.

Thanks in advance and congrats on your helpful website!

## 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** August 9, 2017 at 6:34 am #

REPLY ↩

Even if you frame the sequence problem the same way for LSTMs and MLPs, the units inside each network are different (e.g. LSTMs have memory and gates). In turn, results will probably differ.

I would encourage you to test both types of networks on your problem, and most importantly, brainstorm many different ways to frame your sequence prediction problem to see which works best.

## Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

## 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

## Welcome to Machine Learning Mastery



Hi, I'm Dr. Jason Brownlee.

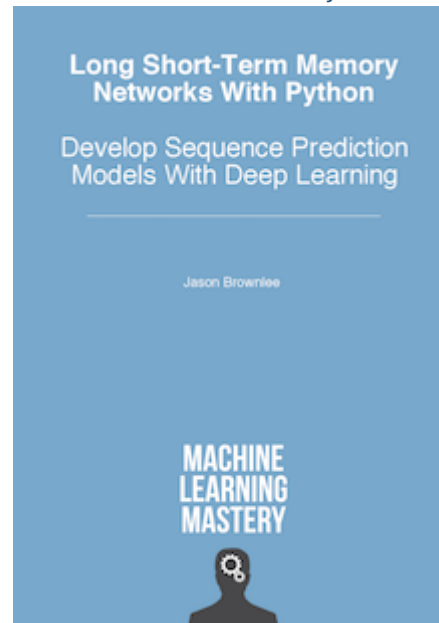
My goal is to make practitioners like YOU awesome at applied machine learning.

[Read More](#)

## Deep Learning for Sequence Prediction

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

Get Started With LSTMs in Python Today!



## 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

POPULAR

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

Get Your Start in Machine Learning



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

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

APRIL 7, 2017

**Multi-Class Classification Tutorial with the Keras Deep Learning Library**

JUNE 2, 2016

**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](#)[Get Your Start in Machine Learning](#)

© 2017 Machine Learning Mastery. All Rights Reserved.

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

## 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