

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

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

# How to Scale Data for Long Short-Term Memory Networks in Python

by **Jason Brownlee** on July 7, 2017 in **Long Short-Term Memory Networks**



The data for your sequence prediction problem probably needs to be scaled when training a neural network, such as a Long Short-Term Memory recurrent neural network.

When a network is fit on unscaled data that has a range of values (e.g. quantities in the 10s to 100s) it is possible for large inputs to slow down the learning and convergence of your network and in some cases prevent the network from effectively learning your problem.

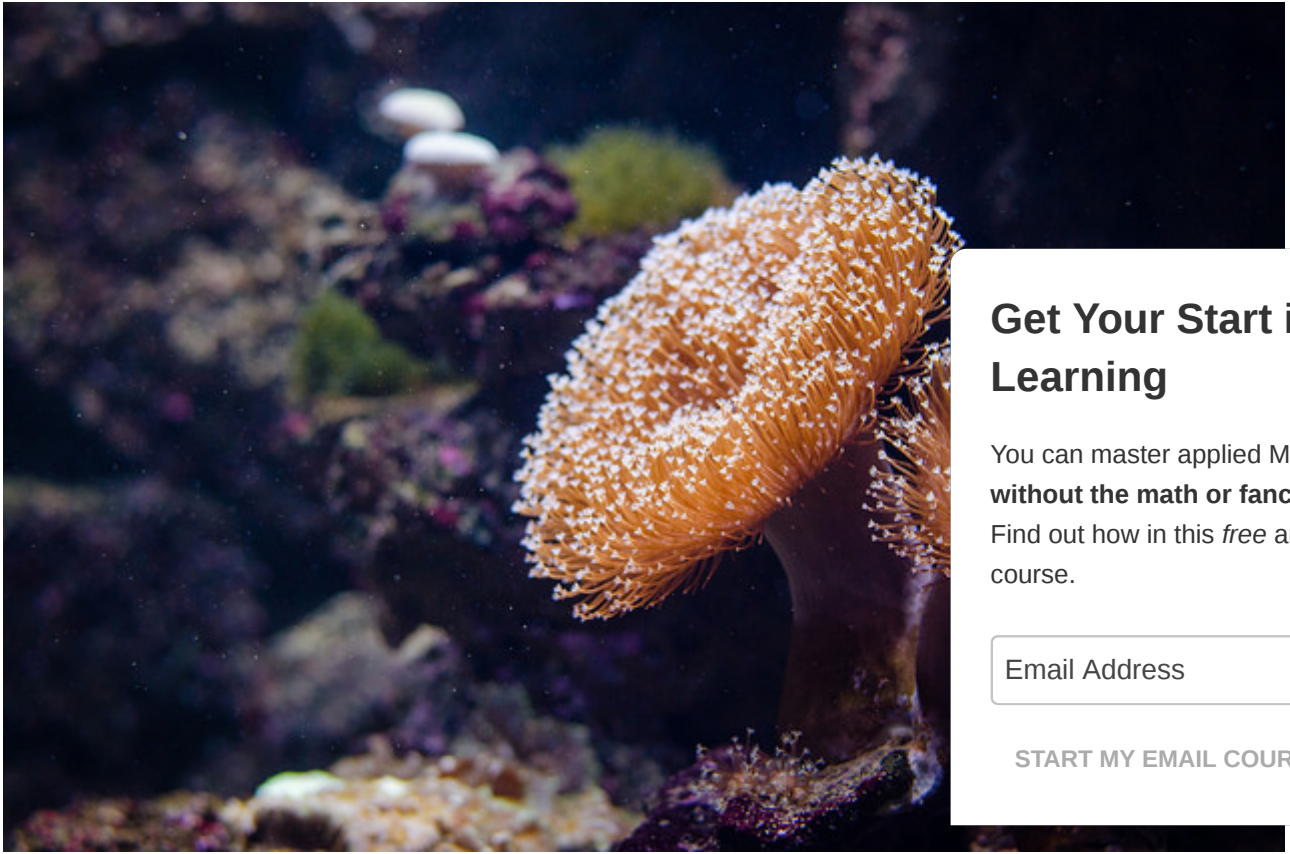
In this tutorial, you will discover how to normalize and standardize your sequence prediction data and how to decide which to use for your input and output variables.

After completing this tutorial, you will know:

[Get Your Start in Machine Learning](#)

- How to normalize and standardize sequence data in Python.
- How to select the appropriate scaling for input and output variables.
- Practical considerations when scaling sequence data.

Let's get started.



### 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 Scale Data for Long Short-Term Memory Networks in Python  
Photo by [Mathias Appel](#), some rights reserved.

## Tutorial Overview

This tutorial is divided into 4 parts; they are:

[Get Your Start in Machine Learning](#)

1. Scaling Series Data
2. Scaling Input Variables
3. Scaling Output Variables
4. Practical Considerations When Scaling

## Scaling Series Data in Python

There are two types of scaling of your series that you may want to consider: normalization and standardization.

These can both be achieved using the scikit-learn library.

### Normalize Series Data

Normalization is a rescaling of the data from the original range so that all values are within the range

Normalization requires that you know or are able to accurately estimate the minimum and maximum of these values from your available data. If your time series is trending up or down, estimating these extremes may not be the best method to use on your problem.

A value is normalized as follows:

```
1 y = (x - min) / (max - min)
```

Where the minimum and maximum values pertain to the value x being normalized.

For example, for a dataset, we could guesstimate the min and max observable values as 30 and -10 as follows:

```
1 y = (x - min) / (max - min)
2 y = (18.8 - (-10)) / (30 - (-10))
3 y = 28.8 / 40
4 y = 0.72
```

You can see that if an x value is provided that is outside the bounds of the minimum and maximum values, that the resulting value will not be in the range of 0 and 1. You could check for these observations prior to making predictions and either remove them from the dataset or limit them to the pre-defined maximum or minimum 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

You can normalize your dataset using the scikit-learn object [MinMaxScaler](#).

Good practice usage with the MinMaxScaler and other scaling techniques is as follows:

- **Fit the scaler using available training data.** For normalization, this means the training data will be used to estimate the minimum and maximum observable values. This is done by calling the `fit()` function.
- **Apply the scale to training data.** This means you can use the normalized data to train your model. This is done by calling the `transform()` function.
- **Apply the scale to data going forward.** This means you can prepare new data in the future on which you want to make predictions.

If needed, the transform can be inverted. This is useful for converting predictions back into their original scale for reporting or plotting. This can be done by calling the `inverse_transform()` function.

Below is an example of normalizing a contrived sequence of 10 quantities.

The scaler object requires data to be provided as a matrix of rows and columns. The loaded time series

```
1 from pandas import Series
2 from sklearn.preprocessing import MinMaxScaler
3 # define contrived series
4 data = [10.0, 20.0, 30.0, 40.0, 50.0, 60.0, 70.0, 80.0, 90.0, 100.0]
5 series = Series(data)
6 print(series)
7 # prepare data for normalization
8 values = series.values
9 values = values.reshape((len(values), 1))
10 # train the normalization
11 scaler = MinMaxScaler(feature_range=(0, 1))
12 scaler = scaler.fit(values)
13 print('Min: %f, Max: %f' % (scaler.data_min_, scaler.data_max_))
14 # normalize the dataset and print
15 normalized = scaler.transform(values)
16 print(normalized)
17 # inverse transform and print
18 inversed = scaler.inverse_transform(normalized)
19 print(inversed)
```

Running the example prints the sequence, prints the min and max values estimated from the sequence, prints the same normalized sequence, then the values back in their original scale using the inverse transform.

We can also see that the minimum and maximum values of the dataset are 10.0 and 100.0 respectively.

## 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

```
1 0      10.0
2 1      20.0
3 2      30.0
4 3      40.0
5 4      50.0
6 5      60.0
7 6      70.0
8 7      80.0
9 8      90.0
10 9     100.0
11
12 Min: 10.000000, Max: 100.000000
13
14 [[ 0.
15   [ 0.11111111]
16   [ 0.22222222]
17   [ 0.33333333]
18   [ 0.44444444]
19   [ 0.55555556]
20   [ 0.66666667]
21   [ 0.77777778]
22   [ 0.88888889]
23   [ 1.
24
25 [[ 10.]
26   [ 20.]
27   [ 30.]
28   [ 40.]
29   [ 50.]
30   [ 60.]
31   [ 70.]
32   [ 80.]
33   [ 90.]
34   [100.]
```

## 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

## Need help with LSTMs for Sequence Prediction?

Take my free 7-day email course and discover 6 different LSTM architectures (with sample code).

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

Get Your Start in Machine Learning

[Start Your FREE Mini-Course Now!](#)

## Standardize Series Data

Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1.

This can be thought of as subtracting the mean value or centering the data.

Like normalization, standardization can be useful, and even required in some machine learning algorithms when your data has input values with differing scales.

Standardization assumes that your observations fit a Gaussian distribution (bell curve) with a well behaved distribution. You should not standardize your time series data if this expectation is not met, but you may not get reliable results.

Standardization requires that you know or are able to accurately estimate the mean and standard deviation of your data. You can estimate these values from your training data.

A value is standardized as follows:

```
1 y = (x - mean) / standard_deviation
```

Where the mean is calculated as:

```
1 mean = sum(x) / count(x)
```

And the standard\_deviation is calculated as:

```
1 standard_deviation = sqrt( sum( (x - mean)^2 ) / count(x))
```

We can guesstimate a mean of 10 and a standard deviation of about 5. Using these values, we can standardize the first value of 20.7 as follows:

```
1 y = (x - mean) / standard_deviation
2 y = (20.7 - 10) / 5
3 y = (10.7) / 5
4 y = 2.14
```

### 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 mean and standard deviation estimates of a dataset can be more robust to new data than the minimum and maximum.

You can standardize your dataset using the scikit-learn object `StandardScaler`.

```
1 from pandas import Series
2 from sklearn.preprocessing import StandardScaler
3 from math import sqrt
4 # define contrived series
5 data = [1.0, 5.5, 9.0, 2.6, 8.8, 3.0, 4.1, 7.9, 6.3]
6 series = Series(data)
7 print(series)
8 # prepare data for normalization
9 values = series.values
10 values = values.reshape((len(values), 1))
11 # train the normalization
12 scaler = StandardScaler()
13 scaler = scaler.fit(values)
14 print('Mean: %f, StandardDeviation: %f' % (scaler.mean_, sqrt(scaler.var_)))
15 # normalize the dataset and print
16 standardized = scaler.transform(values)
17 print(standardized)
18 # inverse transform and print
19 inversed = scaler.inverse_transform(standardized)
20 print(inversed)
```

Running the example prints the sequence, prints the mean and standard deviation estimated from the data, and prints the values back in their original scale.

We can see that the estimated mean and standard deviation were about 5.3 and 2.7 respectively.

```
1 0 1.0
2 1 5.5
3 2 9.0
4 3 2.6
5 4 8.8
6 5 3.0
7 6 4.1
8 7 7.9
9 8 6.3
10
11 Mean: 5.355556, StandardDeviation: 2.712568
12
13 [[-1.60569456]
14 [ 0.05325007]
```

## 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 [ 1.34354035]
16 [-1.01584758]
17 [ 1.26980948]
18 [-0.86838584]
19 [-0.46286604]
20 [ 0.93802055]
21 [ 0.34817357]]
22
23 [[ 1. ]
24 [ 5.5]
25 [ 9. ]
26 [ 2.6]
27 [ 8.8]
28 [ 3. ]
29 [ 4.1]
30 [ 7.9]
31 [ 6.3]]
```

## Scaling Input Variables

The input variables are those that the network takes on the input or visible layer in order to make a prediction.

A good rule of thumb is that input variables should be small values, probably in the range of 0-1 or standard deviation of one.

Whether input variables require scaling depends on the specifics of your problem and of each variable.

## Categorical Inputs

You may have a sequence of categorical inputs, such as letters or statuses.

Generally, categorical inputs are first integer encoded then one hot encoded. That is, a unique integer value is assigned to each distinct possible input, then a binary vector of ones and zeros is used to represent each integer value.

By definition, a one hot encoding will ensure that each input is a small real value, in this case 0.0 or 1.0.

## Real-Valued Inputs

You may have a sequence of quantities as inputs, such as prices or temperatures.

### 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 the distribution of the quantity is normal, then it should be standardized, otherwise the series should be normalized. This applies if the range of quantity values is large (10s 100s, etc.) or small (0.01, 0.0001).

If the quantity values are small (near 0-1) and the distribution is limited (e.g. standard deviation near 1) then perhaps you can get away with no scaling of the series.

## Other Inputs

Problems can be complex and it may not be clear how to best scale input data.

If in doubt, normalize the input sequence. If you have the resources, explore modeling with the raw data, standardized data, and normalized and see if there is a beneficial difference.

“If the input variables are combined linearly, as in an MLP [Multilayer Perceptron], then it is rare that standardizing the inputs is the best choice, at least in theory. ... However, there are a variety of practical reasons why standardizing the inputs can be beneficial, such as the chances of getting stuck in local optima.

— [Should I normalize/standardize/rescale the data?](#) Neural Nets FAQ

## Scaling Output Variables

The output variable is the variable predicted by the network.

You must ensure that the scale of your output variable matches the scale of the activation function (tanh, sigmoid, etc.).

“If your output activation function has a range of  $[0,1]$ , then obviously you must ensure that the target values lie within that range. But it is generally better to choose an output activation function suited to the distribution of the targets than to force your data to conform to the output activation function.

— [Should I normalize/standardize/rescale the data?](#) Neural Nets FAQ

The following heuristics should cover most sequence prediction problems:

### 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

## Binary Classification Problem

If your problem is a binary classification problem, then the output will be class values 0 and 1. This is best modeled with a sigmoid activation function on the output layer. Output values will be real values between 0 and 1 that can be snapped to crisp values.

## Multi-class Classification Problem

If your problem is a multi-class classification problem, then the output will be a vector of binary class values between 0 and 1, one output per class value. This is best modeled with a softmax activation function on the output layer. Again, output values will be real values between 0 and 1 that can be snapped to crisp values.

## Regression Problem

If your problem is a regression problem, then the output will be a real value. This is best modeled with a linear activation function on the output layer. If the output value is normal, then you can standardize the output variable. Otherwise, the output variable can be

## Other Problem

There are many other activation functions that may be used on the output layer and the specifics of y

The rule of thumb is to ensure that the network outputs match the scale of your data.

## Practical Considerations When Scaling

There are some practical considerations when scaling sequence data.

- **Estimate Coefficients.** You can estimate coefficients (min and max values for normalization or mean and standard deviation for standardization) from the training data. Inspect these first-cut estimates and use domain knowledge or domain experts to help improve these estimates so that they will be usefully correct on all data in the future.
- **Save Coefficients.** You will need to normalize new data in the future in exactly the same way as the data used to train your model. Save the coefficients used to file and load them later when you need to scale new data when making predictions.
- **Data Analysis.** Use data analysis to help you better understand your data. For example, a simple histogram can help you quickly get a feeling for the distribution of quantities to see if standardization would make sense.
- **Scale Each Series.** If your problem has multiple series, treat each as a separate variable and in

### 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

- **Scale At The Right Time.** It is important to apply any scaling transforms at the right time. For example, if you have a series of quantities that is non-stationary, it may be appropriate to scale after first making your data stationary. It would not be appropriate to scale the series after it has been transformed into a supervised learning problem as each column would be handled differently, which would be incorrect.
- **Scale if in Doubt.** You probably do need to rescale your input and output variables. If in doubt, at least normalize your data.

## Further Reading

This section lists some additional resources to consider when scaling.

- [Should I normalize/standardize/rescale the data? Neural Nets FAQ](#)
- [MinMaxScaler scikit-learn API documentation](#)
- [StandardScaler scikit-learn API documentation](#)
- [How to Scale Machine Learning Data from Scratch with Python](#)
- [How to Normalize and Standardize Time Series Data in Python](#)
- [How to Prepare Your Data for Machine Learning in Python with Scikit-Learn](#)

## Summary

In this tutorial, you discovered how to scale your sequence prediction data when working with Long Short-Term Memory Networks.

Specifically, you learned:

- How to normalize and standardize sequence data in Python.
- How to select the appropriate scaling for input and output variables.
- Practical considerations when scaling sequence data.

Do you have any questions about scaling sequence prediction data?

Ask your question 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.**

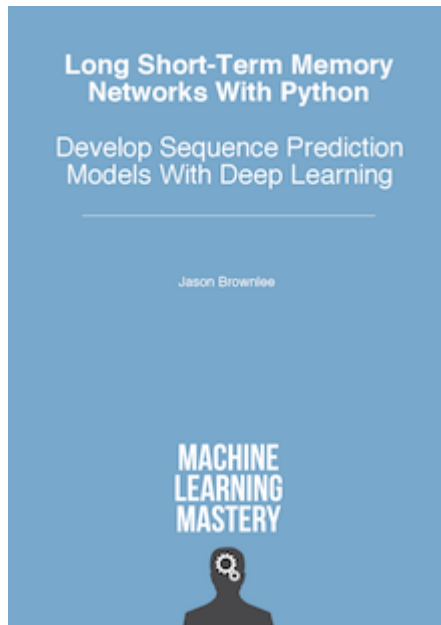
Find out how in this *free* and *practical* email course.

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](#)

### 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](#)



### 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](#) →

[← A Tour of Recurrent Neural Network Algorithms for Deep Learning](#)

[How to Remove Trends and Seasonality with a Difference Transform in Python >](#)

[Get Your Start in Machine Learning](#)

## 11 Responses to *How to Scale Data for Long Short-Term Memory Networks in Python*



**Jack Sheffield** July 7, 2017 at 6:33 am #

REPLY ↩

Thanks for the post Jason, nice and succinct walk-through on how to scale data. I wanted to share a great course on Experfy that covers Machine Learning, especially supervised learning that I've found super helpful in understanding all of this



**Jack Sheffield** July 7, 2017 at 6:34 am #

Here's the link! <https://www.experfy.com/training/courses/machine-learning-foundations-supervised-learning/>



**Jason Brownlee** July 9, 2017 at 10:32 am #

Glad to hear it.



**Anthony The Koala** July 7, 2017 at 9:35 am #

Dear Dr Jason,

When making predictions using the scaled data, do you have to unscale the data, using the

```
1 y (scaled value) = (x-min)/(max-min)
2
3 y * (max - min) + min = actual value
```

OR

```
1 y * std_dev + mean = actual 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

Get Your Start in Machine Learning

Thank you



**Jason Brownlee** July 9, 2017 at 10:34 am #

REPLY ↩

After the prediction, yes, in order to make use if it or have error scores in the correct scale for apples to apples comparison of models.



**Natallia Lundqvist** July 7, 2017 at 10:28 pm #

REPLY ↩

Hi Jason, thank you once again for sharing your great ideas! I work with seq2seq application on text input of variable length with very large vocabulary (several thousand entries). Obviously, padding and one\_hot\_encode are necessary in this case. I use `keras.tokenizer.texts_to_sequences(...)` and then `keras.tokenizer.sequences_to_matrix(sequence, mode='binary')` directly into LSTMs.

For example:

```
seq_test = tokenizer.texts_to_sequences(input_text_sequence)
```

```
seq_test[:4]
```

```
Out[16]: [[1, 2, 110], [23, 5, 150], [1, 3, 17], [8, 2, 218, 332]]
```

```
X_test = tokenizer.sequences_to_matrix(seq_test, mode='binary')
```

```
X_test[:4,:]
```

```
Out[18]:
```

```
array([[ 0.,  1.,  1., ...,  0.,  0.,  0.],
```

```
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
```

```
       [ 0.,  1.,  0., ...,  0.,  0.,  0.],
```

```
       [ 0.,  0.,  1., ...,  0.,  0.,  0.]])
```

If one tries to pass padded sequence into “sequences\_to\_matrix”, an error message is generated:

File “C:....\keras\preprocessing\text.py”, line 262, in sequences\_to\_matrix

if not seq:

ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or a.all()

## 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

Is it so that one has to do “one\_hot\_encoding” manually in order to make use of LSTMs in an encode-decode manner???

On the other hand, I get very good convergence (>99%) if I don't do one\_hot\_encode and use a network architecture similar to <http://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/>

The problem arise with predictions since the last Dense layer has activation='sigmoid', which generates values between 0 and 1. How to make predictions in the form (Out[19]: [[1, 2, 110], [23, 5, 150], [1, 3, 17], [8, 2, 218]]) without one\_hot\_encode input\_sequence???

The last question. If one use one\_hot\_encode of a sequence, embedding layer and convolution layer don't make sense, right???



**Jason Brownlee** July 9, 2017 at 10:45 am #

REPLY ↩

LSTM input is 3D [samples, timesteps, features]. Each sample will be one sequence of you are the one hot encoded 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



**Liviu** July 12, 2017 at 2:09 am #

Hello and thank you for the tutorials ! Learned a lot from them.

One question regarding scaling (or normalization): how can we make sure that the scaling results remain example:

- step 1: we use some data sets to train a model (with scaling data) and then we save the trained model
- step 2: we import the model created at step 1 and used it to predict a prediction data set.

But: the prediction data set must also be scaled. And more than that it must be scaled with the same scale the model trained at step 1. Am I wrong ?

Or we somehow have to save the scaling object also and import it again to be used to scale the prediction data set at step 2 ?



**Jason Brownlee** July 12, 2017 at 9:49 am #

REPLY ↩

Correct.

Get Your Start in Machine Learning

It means you must estimate the scaling parameters carefully and save them for future use.



**Emmanuel** July 31, 2017 at 9:48 am #

REPLY ↩

Thanks for the good work



**Jason Brownlee** July 31, 2017 at 3:48 pm #

REPLY ↩

I'm glad you found the post useful Emmanuel.

## Leave a Reply

Name (required)

Email (will not be published) (required)

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



[SUBMIT COMMENT](#)

## Welcome to Machine Learning Mastery



Hi, I'm Dr. Jason Brownlee.

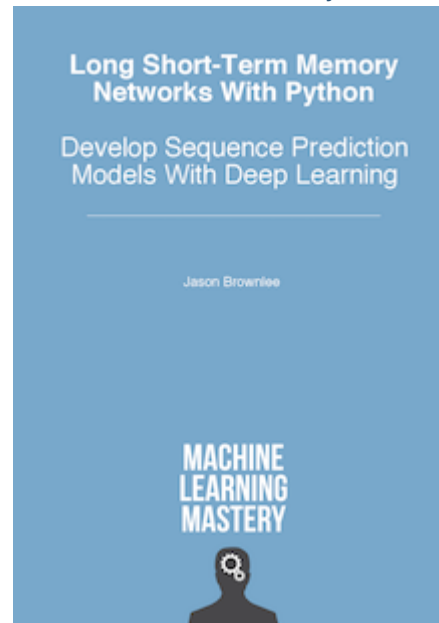
My goal is to make practitioners like YOU awesome at applied machine learning.

[Read More](#)

### Deep Learning for Sequence Prediction

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

Get 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

Get Your Start in Machine Learning

## 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

**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

Get Your Start in Machine Learning