How to Normalize and Standardize Time Series Data in Python

by **Jason Brownlee** on December 12, 2016 in **Time Series**

Some machine learning algorithms will achieve better performance if your time series data has a consistent scale or distribution.

Two techniques that you can use to consistently rescale your time series data are normalization and standardization.

In this tutorial, you will discover how you can apply normalization and standardization rescaling to your time series data in Python.

After completing this tutorial, you will know:

- The limitations of normalization and expectations of your data for using standardization.
- What parameters are required and how to manually calculate normalized and standardized values.
- How to normalize and standardize your time series data using scikit-learn in Python.

Let’s get started.

How to Normalize and Standardize Time Series Data in Python
Photo by Sage Ross, some rights reserved.

Minimum Daily Temperatures Dataset

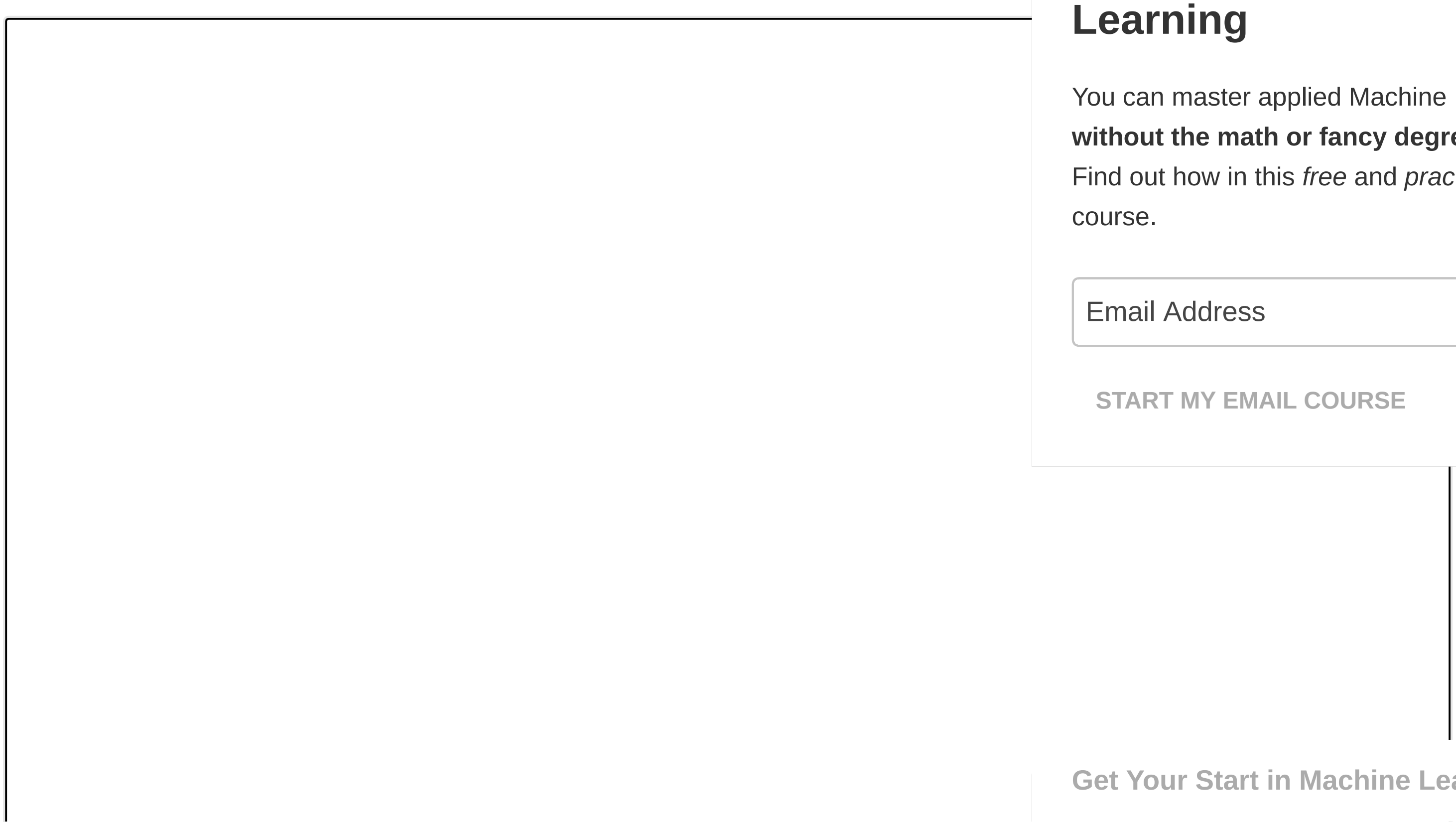
This dataset describes the minimum daily temperatures over 10 years (1981-1990) in the city Melbourne, Australia.

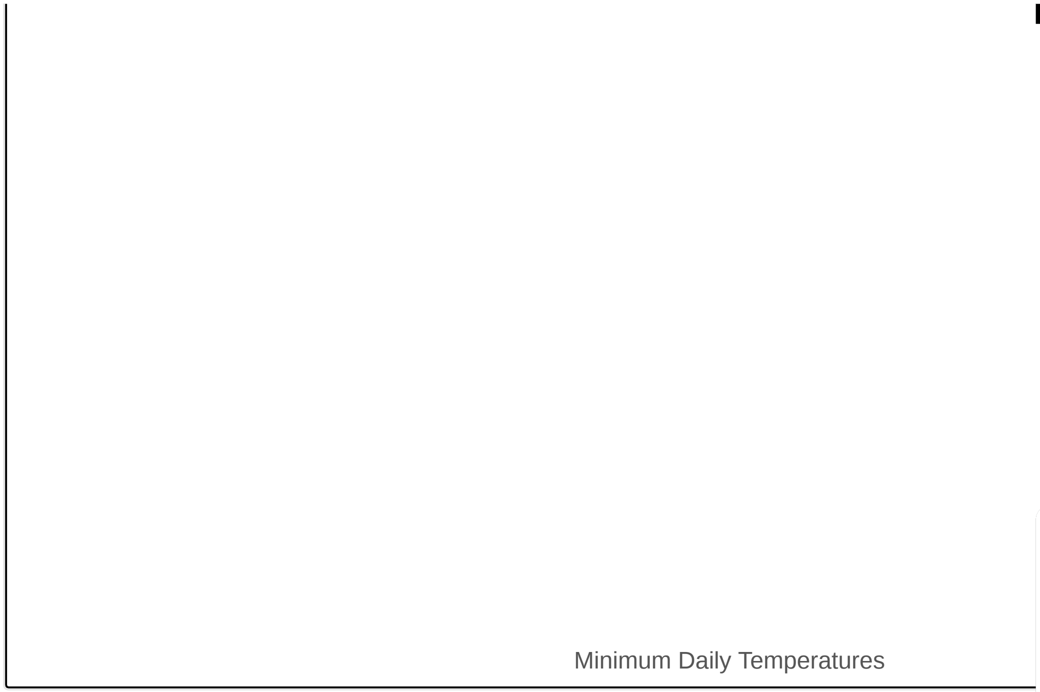
The units are in degrees Celsius and there are 3,650 observations. The source of the data is credited as the Australian Bureau of Meteorology.

Below is a sample of the first 5 rows of data, including the header row.

```
"Date","Temperatures"
"1981-01-01",20.7
"1981-01-02",17.9
"1981-01-03",18.8
"1981-01-04",14.6
"1981-01-05",15.8
```

Below is a plot of the entire dataset taken from Data Market.





The dataset shows a strong seasonality component and has a nice, fine-grained detail to work with.

Download and learn more about the dataset here.

This tutorial assumes that the dataset is in your current working directory with the filename “**daily-mi**

Note: The downloaded file contains some question mark (“?”) characters that must be removed befc
editor and remove the “?” characters. Also remove any footer information in the file.

Start Your FREE Mini-Course Now!

Normalize Time Series Data

Normalization is a rescaling of the data from the original range so that all values are within the range of 0 and 1.

Normalization can be useful, and even required in some machine learning algorithms when your time series data has input values with differing scales.It may be required for algorithms, like k-Nearest neighbors, which uses distance calculations and Linear Regression and Artificial Neural Networks that weight input values.

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

A value is normalized as follows:

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

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

For example, for the temperature data, we could guesstimate the min and max observable values as estimated. We can then normalize any value like 18.8 as follows:

```
y = (x - min) / (max - min)
y = (18.8 - -10) / (30 - -10)
y = 28.8 / 40
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.

You can normalize your dataset using the scikit-learn object MinMaxScaler.

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

1. **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,
2. **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
3. **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 the Minimum Daily Temperatures dataset.

The scaler requires data to be provided as a matrix of rows and columns. The loaded time series dat
reshaped into a matrix of one column with 3,650 rows.

The reshaped dataset is then used to fit the scaler, the dataset is normalized, then the normalization
again.

```python
# Normalize time series data
from pandas import Series
from sklearn.preprocessing import MinMaxScaler
# load the dataset and print the first 5 rows
series = Series.from_csv('daily-minimum-temperatures-in-me.csv', header=0)
print(series.head())
# prepare data for normalization
values = series.values
values = values.reshape((len(values), 1))
# train the normalization
scaler = MinMaxScaler(feature_range=(0, 1))
scaler = scaler.fit(values)
print('Min: %f, Max: %f' % (scaler.data_min_, scaler.data_max_))
# normalize the dataset and print the first 5 rows
normalized = scaler.transform(values)
for i in range(5):
    print(normalized[i])
# inverse transform and print the first 5 rows
inversed = scaler.inverse_transform(normalized)
for i in range(5):
    print(inversed[i])
```

Running the example prints the first 5 rows from the loaded dataset, shows the same 5 values in the [redacted] original scale using the inverse transform.

We can also see that the minimum and maximum values of the dataset are 0 and 26.3 respectively.

```
Date
1981-01-01 20.7
1981-01-02 17.9
1981-01-03 18.8
1981-01-04 14.6
1981-01-05 15.8
Name: Temp, dtype: float64
Min: 0.000000, Max: 26.300000
[ 0.78707224]
[ 0.68060837]
[ 0.7148289]
[ 0.55513308]
[ 0.60076046]
[ 20.7]
[ 17.9]
[ 18.8]
[ 14.6]
[ 15.8]
```

There is another type of rescaling that is more robust to new values being outside the range of expe
look at that next.

Standardize Time Series Data

Standardizing a dataset involves rescaling the distribution of values so that the mean of observed va

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 time series data has input values with differing scales.

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

This includes algorithms like Support Vector Machines, Linear and Logistic Regression, and other al[redacted]ce with Gaussian data.

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

A value is standardized as follows:

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

Where the *mean* is calculated as:

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

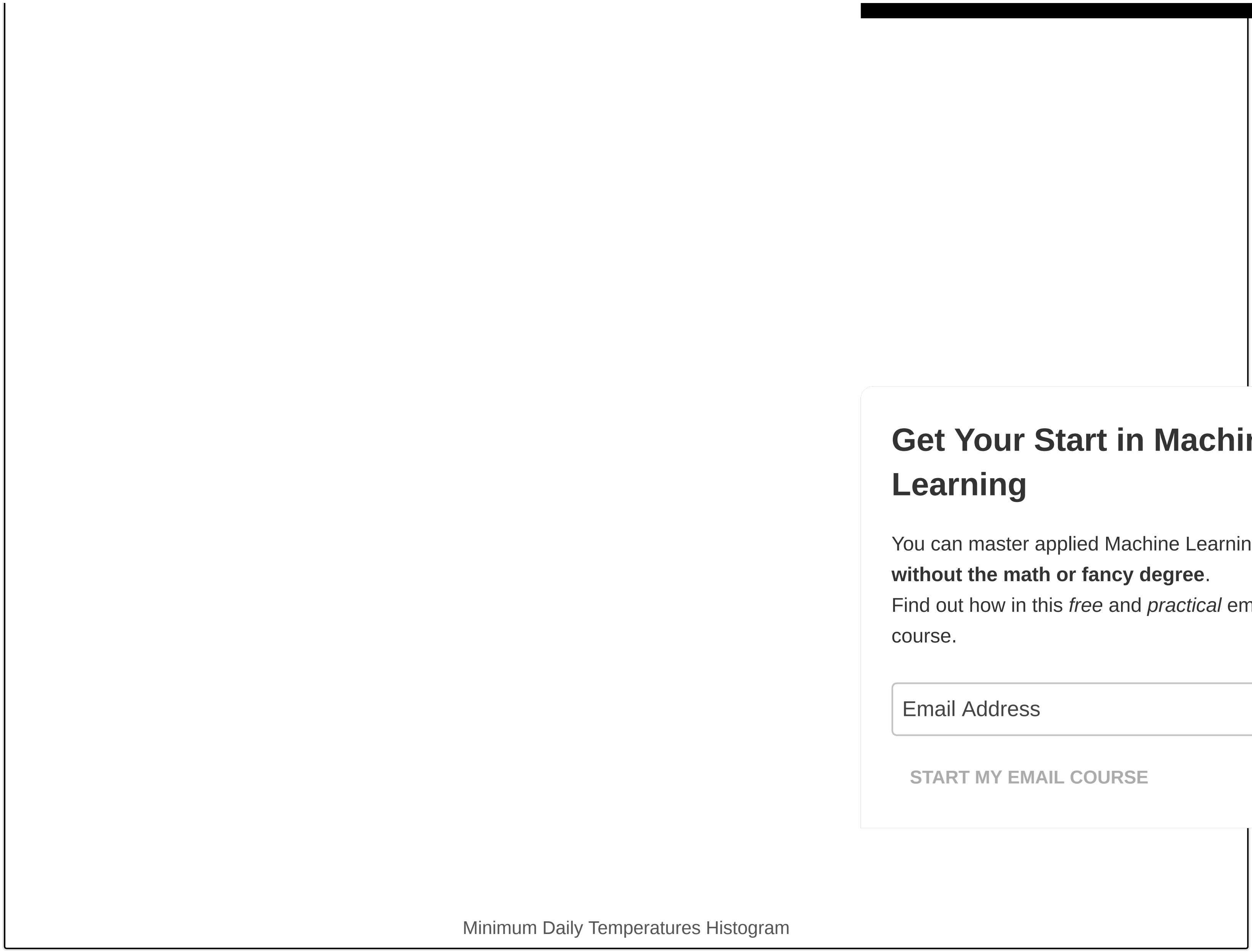
And the *standard_deviation* is calculated as:

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

For example, we can plot a histogram of the Minimum Daily Temperatures dataset as follows:

```
from pandas import Series
from matplotlib import pyplot
series = Series.from_csv('daily-minimum-temperatures-in-me.csv', header=0)
series.hist()
pyplot.show()
```

Running the code gives the following plot that shows a Gaussian distribution of the dataset, as assu



Minimum Daily Temperatures Histogram

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

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

```
y = (20.7 - 10) / 5
y = (10.7) / 5
y = 2.14
```

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.

Below is an example of standardizing the Minimum Daily Temperatures dataset.

```
# Standardize time series data
from pandas import Series
from sklearn.preprocessing import StandardScaler
from math import sqrt
# load the dataset and print the first 5 rows
series = Series.from_csv('daily-minimum-temperatures-in-me.csv', header=0)
print(series.head())
# prepare data for standardization
values = series.values
values = values.reshape((len(values), 1))
# train the standardization
scaler = StandardScaler()
scaler = scaler.fit(values)
print('Mean: %f, StandardDeviation: %f' % (scaler.mean_, sqrt(scaler.var_)))
# standardization the dataset and print the first 5 rows
normalized = scaler.transform(values)
for i in range(5):
    print(normalized[i])
# inverse transform and print the first 5 rows
inversed = scaler.inverse_transform(normalized)
for i in range(5):
    print(inversed[i])
```

Running the example prints the first 5 rows of the dataset, prints the same values standardized, then

We can see that the estimated mean and standard deviation were 11.1 and 4.0 respectively.

```
Date
1981-01-01 20.7
1981-01-02 17.9
1981-01-03 18.8
1981-01-04 14.6
1981-01-05 15.8
Name: Temp, dtype: float64
```

```
Mean: 11.177753, StandardDeviation: 4.071279
[ 2.33888328]
[ 1.65113873]
[ 1.87219948]
[ 0.84058266]
[ 1.13533032]
[ 20.7]
[ 17.9]
[ 18.8]
[ 14.6]
[ 15.8]
```

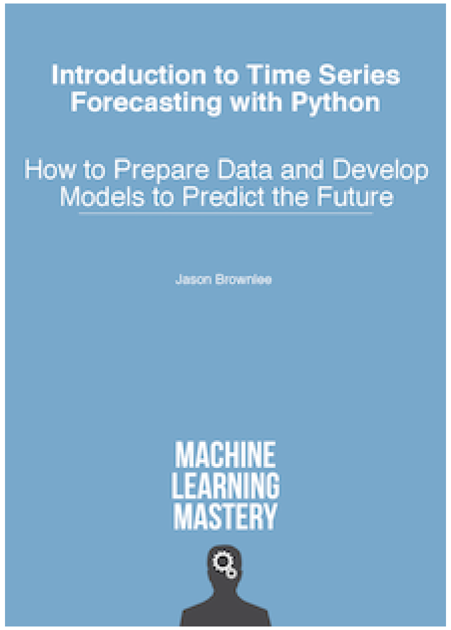
Summary

In this tutorial, you discovered how to normalize and standardize time series data in Python.

Specifically, you learned:

- That some machine learning algorithms perform better or even require rescaled data when mod
- How to manually calculate the parameters required for normalization and standardization.
- How to normalize and standardize time series data using scikit-learn in Python.

Do you have any questions about rescaling time series data or about this post?
Ask your questions in the comments and I will do my best to answer.


Introduction to Time Series Forecasting with Python
How to Prepare Data and Develop Models to Predict the Future
Jason Brownlee
MACHINE LEARNING MASTERY

About Jason Brownlee

Dr. Jason Brownlee is a husband, proud father, academic researcher, author, professional devel
to helping developers get started and get good at applied machine learning. Learn more.
View all posts by Jason Brownlee →

< How to Load and Explore Time Series Data in Python

Basic Feature Engineering With Time Series Data in Python >

39 Responses to *How to Normalize and Standardize Time Series Data in Python*

Marek December 13, 2016 at 8:48 am # REPLY

I assume that this works like a treat for data sets that you can fit into memory … But what about very large data sets that simply would never fit into a single machine. Would you recommend other techniques?

Jason Brownlee December 14, 2016 at 8:22 am #

Great question Marek.

I would suggest estimating the parameters required (min/max for normalization and mean/stdev for s
prepare data just in time prior to use in a model.

Does that help?

Gonzalo December 14, 2016 at 10:06 am #

IMO

If you have kind of stream data, you need to define a range of data to evaluate.

If you just have distributed data, you need to mapreduce.

Fabio December 14, 2016 at 5:53 am # REPLY

Hello Jason,

thank you for your example. I am learning Python und Pandas. Why do you need to reshape the Series.values?

```
# prepare data for standardization
values = series.values
values = values.reshape((len(values), 1))
```

Bye

Jason Brownlee December 14, 2016 at 8:29 am #

REPLY

Great question, it's because the sklearn tools prefer a 2D matrix and the series is 1D.

We just need to be explicit in the numpy array about the number of rows and cols and sklearn will th

Does that help?

Fabio December 15, 2016 at 12:26 am #

Yep! Thank you Jason 🙂

Barnett December 15, 2016 at 8:40 am #

In relation to this topic, how do you usually handle variables of mixed types (e.g. a mixture of categorical, continuous, ordinal variables) in a classifier (e.g. logistic regression, SVM, etc.)? I first perform dummy coding on categorical variables, followed by mixing them with the other variables (after normalizing them to [0, 1]); not sure if this is the best practice. On the other hand, the same question for applying clustering algorithms (say, k-means, spectral clusterings). Thank you.

Jason Brownlee December 16, 2016 at 5:34 am #

REPLY

Hi Barnett, yes exactly as you describe.

I try integer encodings if there is an ordinal relationship.

For categorical variables, I use dummy (binary) variables.

I try to make many different views/perspectives of a prediction problem, including transforms, projections and feature selection filters. I then test them all on a suite of methods and see which representations are generally better at exposing the structure of the problem. While that is running, I do the traditional careful analysis, but this automated method is often faster and results in non-intuitive results.

Magnus January 5, 2017 at 2:02 am # REPLY

I was not able to run this using the data set as is. In the csv file, there is a footer with 3 columns
after removing this and replacing it works)

Jason Brownlee January 5, 2017 at 9:23 am #

Thanks for the tip Magnus.

Yes, the tutorial does assume a well-formed CSV file.

A raw download from DataMarket does contain footer info that must be deleted.

Kensu January 12, 2017 at 1:30 am # REPLY

What is the mathmatical function to denormalize if the function
y = (x – min) / (max – min) is our normalize function.

sevenless January 31, 2017 at 8:53 pm #

[redacted]

Thank you for the nice tutorial.
I wonder how you would normalize the standard deviation for replicate measurements?
Let's assume that we have three measurements for each day instead of only one and that you would want to plot the temperature normalized to its mean as a time series for a single month. Would the standard deviation for each day have to be normalized as well?

Jason Brownlee February 1, 2017 at 10:49 am # REPLY

Great question,

Generally, this is a problem specific question and you can choose the period over which to standardize or normalize.

I would prefer to pick min/max or mean/stdev that are suitable for the entire modeling period per vari

Try other approaches, and see how they fair on your dataset.

Magnus February 17, 2017 at 3:24 am #

Let's say I have a time series and normalize the data in the range 0,1. I train the model and run
occur with values higher than the max value in my training set. The prediction for that event might then s
observation. How to deal with this?

I suppose one possibility is to use e.g. extreme event analysis to estimate a future max value and use th
my training data will be in a narrower range, e.g. 0 to 0.9. Of course, I can do this anyway without an ana
extreme weather phenomena or earthquakes etc.

How is it possible to forecast, accurately, an extreme event, when we don't have this in the training set? After all, extreme events are often very important to be able to forecast.

Jason Brownlee February 17, 2017 at 9:56 am # REPLY

Great question Magnus.

This is an important consideration when scaling.

Standardization will be more robust. Normalization will require you to estimate the limits of expected values, to detect when new input data exceeds those limits and handle that accordingly (report an error, clip, issue a warning, re-train the model with new limits, etc.).

As for the "best" thing to do, that really depends on the domain and your project requirements.

Magnus February 20, 2017 at 10:41 pm # REPLY

What about if the data is highly asymmetric with a negative (or positive) skew, and therefore far from being Gaussian?

If I choose a NN, I assume that my data should be normalised. If I standardise the data it will still be skewed, so when using a NN is it better to transform the data to remove the skew? Or is neural networks a bad choice with skewed data?

Jason Brownlee February 21, 2017 at 9:36 am #

Consider a power transform like a box-cox to make the data more Gaussian, then s

Seine Yumnam April 2, 2017 at 6:36 am #

I don't think this way of scaling time series works. For instance, the standardization method in p
using the whole data set you provide. But in reality, we won't have that. As a result, scaling this way will
data to calculate the mean and std. So we need to figure out a way to calculate the mean and the std based on the data we have at a given point in time.

Jason Brownlee April 4, 2017 at 9:04 am # REPLY

Yes, you might be better off estimating the coefficients needed for scaling based on domain knowledge.

DharaPJ April 4, 2017 at 8:18 pm # REPLY

What if the code shows "NO module named sklearn.preprocessing". I already downloaded scipy 0.18.1 version.

Jason Brownlee April 9, 2017 at 2:30 pm # REPLY

It sounds like sklearn is not installed.

Riccardo April 5, 2017 at 1:43 am #

Hi Jason, thanks for your great work.
I have a question: how to properly normalize train, validation and test dataset for trading forex? Those da
before validation that is before test. I don't want to use future data for normalization since i don't have to
Now my results using Reinforcement Learning are not great but i think that great part of the work is done
My strategy now is doing standardization for every episode(one week of data) of training/test/validation v .

Jason Brownlee April 9, 2017 at 2:31 pm #

I would recommend estimating the min/max or mean/stdev from training data and domain k

Riccardo April 20, 2017 at 1:32 am # REPLY

Thanks for the reply. It's exactly what i'm doing now. Probably there is no other standard possibility than domain knowledge. Would be interesting to find something standard but it seems impossible.

REPLY

Seo Young Jae April 24, 2017 at 12:45 pm #

Thank you for good information!

And.. Can i standardize & normalize a nominal(categorical) variable?

Jason Brownlee April 25, 2017 at 7:41 am #

REPLY

I'm glad you found it useful.

You can encode a nominal variable as integers and then use a one hot encoding.

GEORGIOS PLIGOROPOULOS June 15, 2017 at 11:02 am #

What if all my time series have a trend upwards. Or even worse what if half of my time series ha
trend downwards?

Would detrending the time series be helpful? ("removing" the slope and subtracting the mean)
If yes, what kind of detrending would be useful? Should I detrend each sequence separately from all the
in groups?

But this does not seem like normalization/standardization because you cannot revert the process if you
information.

What are your thoughts on this approach? Thank you

Jason Brownlee June 16, 2017 at 7:45 am #

REPLY

Yes, treat each series separately, make each stationary.

Use differencing:
http://machinelearningmastery.com/difference-time-series-dataset-python/

Marianico July 12, 2017 at 10:51 pm # REPLY

But if you normalize by this way, you are using information from future, therefore the model will overfit. Let's say you normalize the columns to train my model with the mean and the std of their content, but a new input data cannot be normalized following the old criteria, and neither the new one (the mean and the std of the last N rows) because of the trend, or the std…

What about using this instead: http://scikit-learn.org/stable/modules/preprocessing.html#custom-transformers

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

Yes, I would recommend estimating the normalization coefficients (min/max) using training

Marianico July 13, 2017 at 8:15 pm #

Ok, that makes sense. But even with your training data you'd have the same problem: v
normalize/scale using the min/max of whole training dataset, you are taking into account values
won't have this information, right?

Moreover, how would you normalize/scale future data? Using the same saved preprocessing model of your training data, or creating a new MinMaxScaler() using the last N rows? What if the new values are slightly different of the training ones?

That's why I've posted the log1p solution, which is the same as log(1+x) and which will thus work on (-1;∞). Or what about this one: http://timvieira.github.io/blog/post/2014/02/11/exp-normalize-trick/

I think would be very interesting that you pointing this out in the post because I'm afraid it can affect dramatically to model's accuracy…

Jason Brownlee July 14, 2017 at 8:27 am # REPLY

This was my point. If normalizing you need to select min/max values based on available data and domain knowledge (to guestimate the expected max/min that will ever be seen).

Same idea with estimating the mean/stdev for standardization.

If evaluating a model, these estimates should be drawn from the training data only and/or domain expertise.

I hope that is clearer.

Marianico July 13, 2017 at 9:27 pm #

I think the best approach would be the following:

```
scaler = StandardScaler() # or MinMaxScaler()

scaler_train = scaler.fit(X_train)
X_train = scaler_train.transform(X_train)

scaler_full = scaler.fit(X) # X_train + X_test
X_test = scaler_full.transform(X_test)
```

Further reading:

https://www.researchgate.net/post/If_I_used_data_normalization_x-meanx_stdx_for_training_data_would_I_use_train_Mean_and_Standard_Deviation_to_normalize_test_
https://stats.stackexchange.com/questions/267012/difference-between-preprocessing-train-and-test-set-before-and-after-splitting/270284
http://sebastianraschka.com/Articles/2014_about_feature_scaling.html

pranav July 14, 2017 at 5:30 am # REPLY

Hi Jason, thank you for your post.
I have question.

I have timestamp and system-up-time where up-time is # of hrs system has been up in its lifetime. [redacted]
Now I have to predict system failure based on the age of the system or system-up-time hrs.
# of failures might grow based on the age of the system or how many hrs its been running.
I have limited training data and the max up-time hrs in the training data is 1,000 hrs and age is 1,200 hrs. But in real time it could go beyond 100,000 hrs and age could go beyond 150,000 hrs.
How do I standardize timestamp and up-time hrs.

Jason Brownlee July 14, 2017 at 8:35 am #

REPLY

Consider looking into survival analysis:
https://en.wikipedia.org/wiki/Survival_analysis

Kyana August 23, 2017 at 7:17 pm #

Hi Jason,

Thank you for your comprehensive explanation. I have a noisy time series with missing data and outliers standardization works in my case? My sample size can be quite big. Looking forward to your feedback.

Jason Brownlee August 24, 2017 at 6:28 am #

Hmm, perhaps not. But I generally recommend testing and getting data rather than using opinions/advice. I'm often wrong.

Perhaps try to patch the missing data and trim the outliers as secondary steps and see if that impacts model skill.

Let me know how you go.

Sebastian August 23, 2017 at 7:45 pm #

Hi! I did not manage to max the MinMaxScaler work for my tensors of rank 5. Someone knows [redacted] guess you could flatten it scale it and then reshape it back. but I prefer not to get lost with all the dimensions. What I did for now is to make my own normalize to scale numpy tensors if someone bumps into the same problem.

class normalize():

def fit(self, train, interval=(0,1)):
self.min, self.max = train.min(), train.max()
self.interval = interval

return self

def transform(self, train, val, test):

def trans(x):
y = ((self.interval[1]-self.interval[0])*x + \
(self.interval[0]*self.max-self.interval[1]*self.min)) / \
(self.max-self.min)
return y

train_norm = trans(train)
val_norm = trans(val)
test_norm = trans(test)

return train_norm, val_norm, test_norm

def inverse_transform(self, train_norm, val_norm, test_norm):

def inv_trans(y):
x = ((self.max-self.min)*y + \
(self.interval[1]*self.min-self.interval[0]*self.max)) / \
(self.interval[1]-self.interval[0])
return x

train = inv_trans(train_norm)
val = inv_trans(val_norm)
test = inv_trans(test_norm)

return train, val, test

Jason Brownlee August 24, 2017 at 6:29 am # REPLY

The sklearn tools will apply across all columns of your data.

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

Welcome to Machine Learning Mastery