How To Backtest Machine Learning Models for Time Series Forecasting

by **Jason Brownlee** on December 19, 2016 in **Time Series**

k-fold Cross Validation Does Not Work For Time Series Data and Techniques That You Can Use Instead.

The goal of time series forecasting is to make accurate predictions about the future.

The fast and powerful methods that we rely on in machine learning, such as using train-test splits and k-fold cross validation, do not work in the case of time series data. This is because they ignore the temporal components inherent in the problem.

In this tutorial, you will discover how to evaluate machine learning models on time series data with Python. In the field of time series forecasting, this is called backtesting or hindcasting.

After completing this tutorial, you will know:

- The limitations of traditional methods of model evaluation from machine learning and why evaluating models on out of sample data is required.
- How to create train-test splits and multiple train-test splits of time series data for model evaluation in Python.
- How walk-forward validation provides the most realistic evaluation of machine learning models on time series data.

Let’s get started.

How To Backtest Machine Learning
Photo by Nasa, sor

Model Evaluation

How do we know how good a given model is?

We could evaluate it on the data used to train it. This would be invalid. It might provide insight into how the selected model works, and even how it may be improved. But, any estimate of performance on this data would be optimistic, and any decisions based on this performance would be biased.

Why?

It is helpful to take it to an extreme:

> **A model that remembered the timestamps and value for each observation would achieve perfect performance.**

All real models we prepare will report a pale version of this result.

When evaluating a model for time series forecasting, we are interested in the performance of the model on data that was not used to train it. In machine learning, we call this unseen or out of sample data.

We can do this by splitting up the data that we do have available. We use some to prepare the model and we hold back some data and ask the model to make predictions for that period. The evaluation of these predictions will provide a good proxy for how the mode

In applied machine learning, we often split our data into a train and a test set: the training set used to prepare the model and the test set used to evaluate it. We may even use k-fold cross validation that repeats this process by systematically splitting the data into k groups, each given a chance to be a held out model.

These methods cannot be directly used with time series data.

This is because they assume that there is no relationship between the observations, that each observation is independent.

This is not true of time series data, where the time dimension of observations means that we cannot randomly split them into groups. Instead, we must split data up and respect the temporal order in which values were observed.

In time series forecasting, this evaluation of models or e series domains, such as meteorology, this is called hir

We will look at three different methods that you can us ne series problems. They are:

1. **Train-Test split** that respect temporal order of ob
2. **Multiple Train-Test splits** that respect temporal
3. **Walk-Forward Validation** where a model may be

First, let's take a look at a small, univariate time series three backtesting methods: the Sunspot dataset.

Monthly Sunspot Dataset

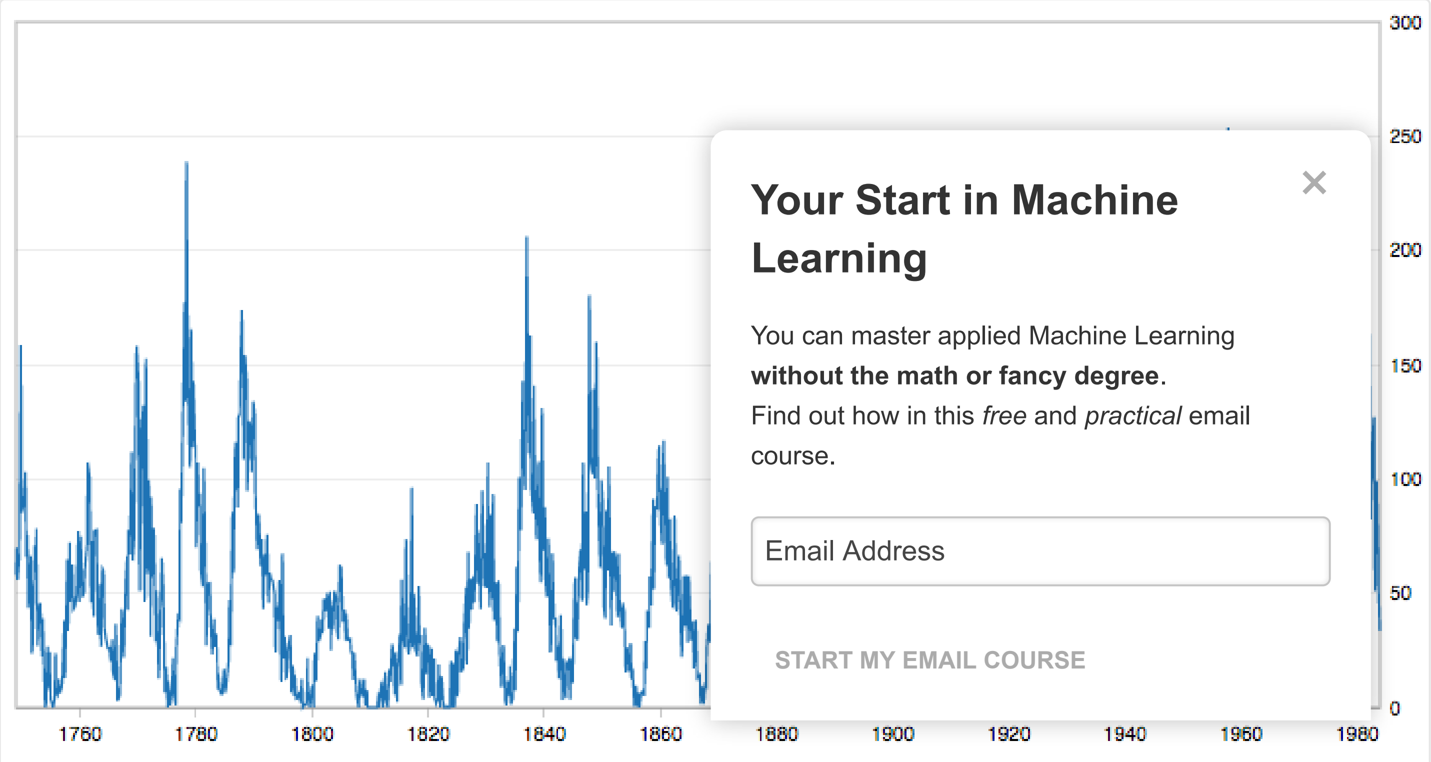
This dataset describes a monthly count of the number of observed sunspots for just over 230 years (1749-1983).

The units are a count and there are 2,820 observations. The source of the dataset is credited as Andrews & Herzberg (1985).

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

```
"Month","Sunspots"
"1749-01",58.0
"1749-02",62.6
"1749-03",70.0
"1749-04",55.7
"1749-05",85.0
```

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



Monthly Sunspot Dataset

The dataset shows seasonality with large differences between seasons.

Download and learn more about the dataset here.

Download the dataset and save it into your current working directory with the filename “*sunspots.csv*“.

Load Sunspot Dataset

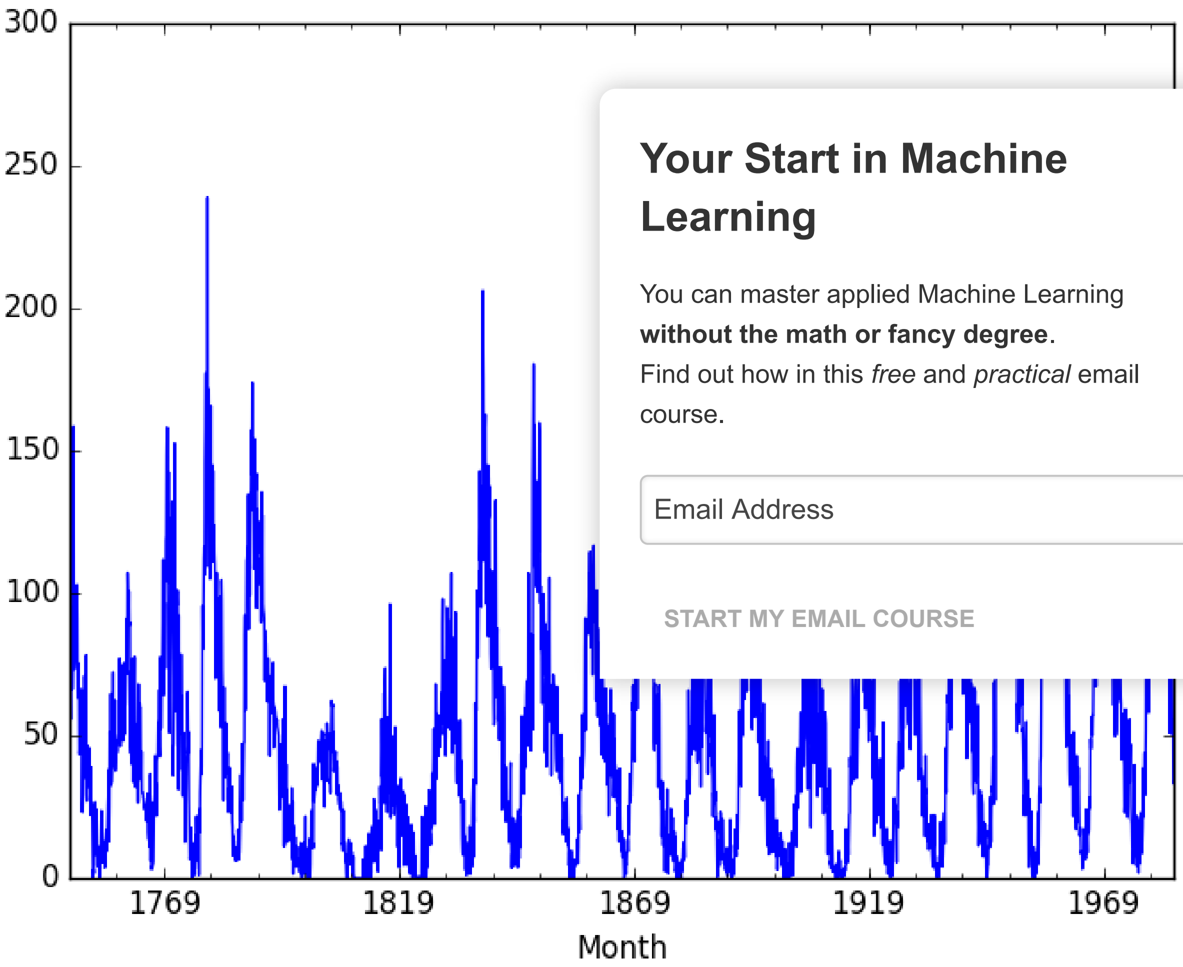
We can load the Sunspot dataset using Pandas.

```
# Load sunspot data
from pandas import Series
from matplotlib import pyplot
series = Series.from_csv('sunspots.csv', header=0)
print(series.head())
series.plot()
pyplot.show()
```

Running the example prints the first 5 rows of data.

```
Month
1749-01-01 00:00:00 58.0
1749-02-01 00:00:00 62.6
1749-03-01 00:00:00 70.0
1749-04-01 00:00:00 55.7
1749-05-01 00:00:00 85.0
Name: Sunspots, dtype: float64
```

The dataset is also plotted.



Plot of the Sunspot Dataset

Train-Test Split

You can split your dataset into training and testing subsets.

Your model can be prepared on the training dataset and predictions can be made and evaluated for the test dataset.

This can be done by selecting an arbitrary split point in the ordered list of observations and creating two new datasets. Depending on the amount of data you have available and the amount of data required, you can use splits of 50-50, 70-30 and 90-10.

It is straightforward to split data in Python.

After loading the dataset as a Pandas Series, we can extract the NumPy array of data values. The split point can be calculated as a specific index in the array. All records up to the split point are taken as the training dataset and all records from the split point to the end of the list of observations are taken as the test set.

Below is an example of this in Python using a split of 66-34.

```
from pandas import Series
series = Series.from_csv('sunspots.csv', header=0)
X = series.values
train_size = int(len(X) * 0.66)
train, test = X[0:train_size], X[train_size:le
print('Observations: %d' % (len(X)))
print('Training Observations: %d' % (len(train
print('Testing Observations: %d' % (len(test))
```

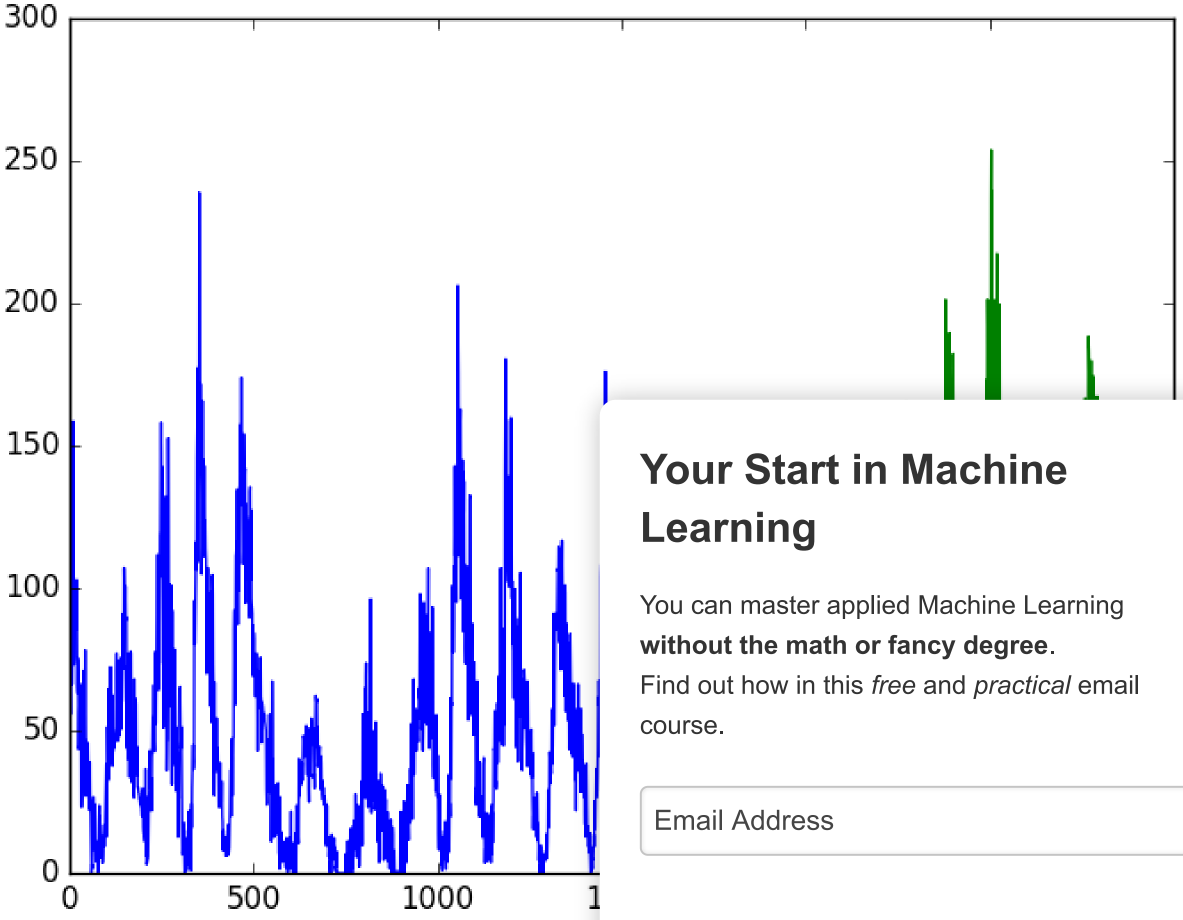
Running the example prints the size of the loaded dat
from the split.

```
Observations: 2820
Training Observations: 1861
Testing Observations: 959
```

We can make this visually by plotting the training and

```
from pandas import Series
from matplotlib import pyplot
series = Series.from_csv('sunspots.csv', head
X = series.values
train_size = int(len(X) * 0.66)
train, test = X[0:train_size], X[train_size:len(X)]
print('Observations: %d' % (len(X)))
print('Training Observations: %d' % (len(train)))
print('Testing Observations: %d' % (len(test)))
pyplot.plot(train)
pyplot.plot([None for i in train] + [x for x in test])
pyplot.show()
```

Running the example plots the training dataset as blue and the test dataset as green.



Sunspot Dataset Train-Test Split

Using a train-test split method to evaluate machine learning models is fast. Preparing the data is simple and intuitive and only one model is created and evaluated.

It is useful when you have a large amount of data so that both training and tests sets are representative of the original problem.

Next, we will look at repeating this process multiple times.

Multiple Train-Test Splits

We can repeat the process of splitting the time series into train and test sets multiple times.

This will require multiple models to be trained and evaluated, but this additional computational expense will provide a more robust estimate of the expected performance of the chosen method and configuration on unseen data.

We could do this manually by repeating the process described in the previous section with different split points.

Alternately, the scikit-learn library provides this capability for us in the *TimeSeriesSplit* object.

You must specify the number of splits to create and the *TimeSeriesSplit* to return the indexes of the train and test observations for each requested split.

The total number of training and test observations are calculated each split iteration (*i*) as follows:

```
training_size = i * n_samples / (n_splits + 1) + n_samples % (n_splits + 1)
test_size = n_samples / (n_splits + 1)
```

Where *n_samples* is the total number of observations and *n_splits* is the total number of splits.

Let's make this concrete with an example. Assume we have 100 observations and we want to create 2 splits.

For the first split, the train and test sizes would be calc

```
train = i * n_samples / (n_splits + 1) + n_sam
train = 1 * 100 / (2 + 1) + 100 % (2 + 1)
train = 33.3 or 33

test = n_samples / (n_splits + 1)
test = 100 / (2 + 1)
test = 33.3 or 33
```

Or the first 33 records are used for training and the ne

The second split is calculated as follows:

```
train = i * n_samples / (n_splits + 1) + n_sam
train = 2 * 100 / (2 + 1) + 100 % (2 + 1)
train = 66.6 or 67

test = n_samples / (n_splits + 1)
test = 100 / (2 + 1)
test = 33.3 or 33
```

Or, the first 67 records are used for training and the remaining 33 records are used for testing.

You can see that the test size stays consistent. This means that performance statistics calculated on the predictions of each trained model will be consistent and can be combined and compared. It provides an apples-to-apples comparison.

What differs is the number of records used to train the model each split, offering a larger and larger history to work with. This may make an interesting aspect of the analysis of results. Alternately, this too could be controlled by holding the number of observations used to train the model consistent and only using the same number of the most recent (last) observations in the training dataset each split to train the model, 33 in this contrived example.

Let's look at how we can apply the TimeSeriesSplit on our sunspot data.

The dataset has 2,820 observations. Let's create 3 splits for the dataset. Using the same arithmetic above, we would expect the following train and test splits to be created:

- **Split 1**: 705 train, 705 test
- **Split 2**: 1,410 train, 705 test
- **Split 3**: 2,115 train, 705 test

As in the previous example, we will plot the train and test observations using separate colors. In this case, we will have 3 splits, so that will be 3 separate plots of the data.

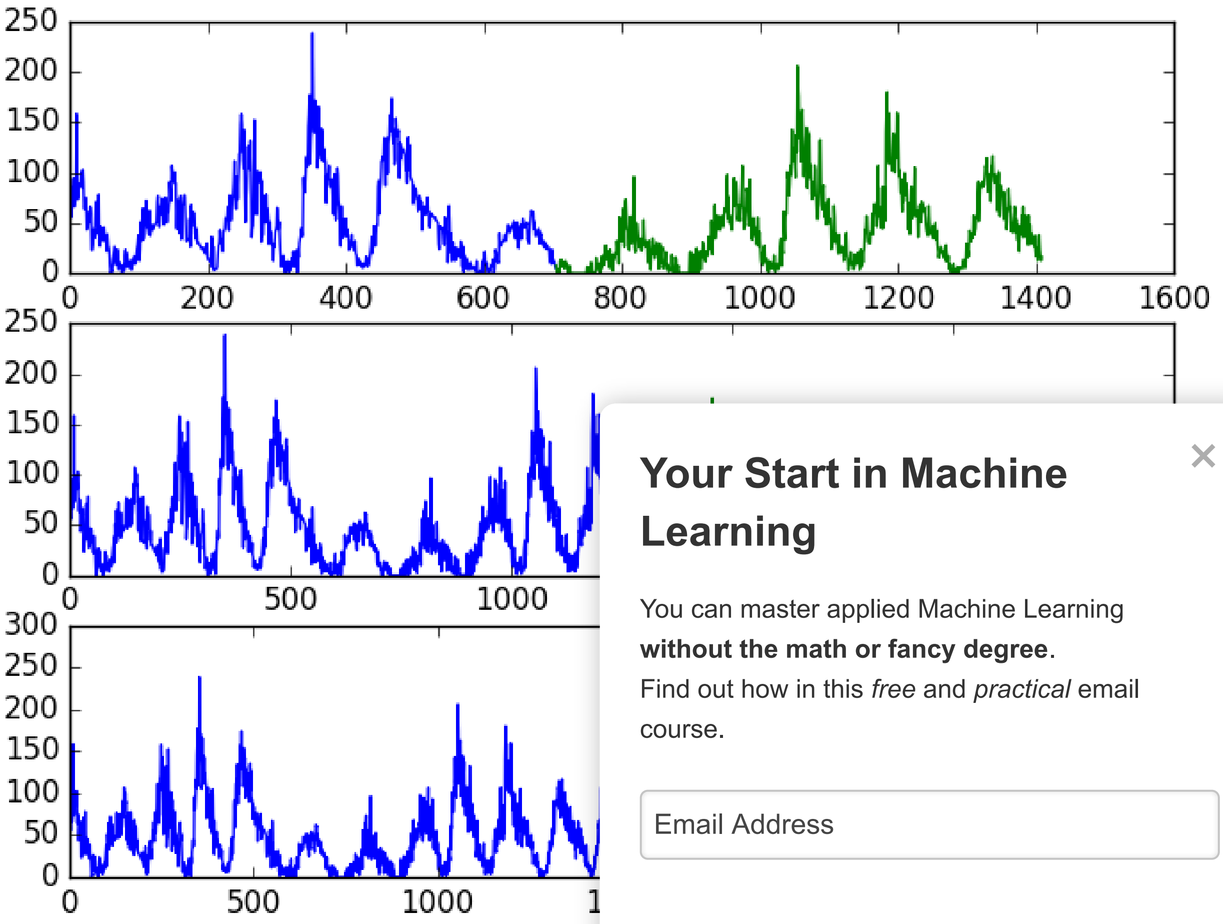
```
from pandas import Series
from sklearn.model_selection import TimeSeriesSplit
from matplotlib import pyplot
series = Series.from_csv('sunspots.csv', head
X = series.values
splits = TimeSeriesSplit(n_splits=3)
pyplot.figure(1)
index = 1
for train_index, test_index in splits.split(X
    train = X[train_index]
    test = X[test_index]
    print('Observations: %d' % (len(train) +
    print('Training Observations: %d' % (len(
    print('Testing Observations: %d' % (len(t
    pyplot.subplot(310 + index)
    pyplot.plot(train)
    pyplot.plot([None for i in train] + [x fo
    index += 1
pyplot.show()
```

Running the example prints the number and size of th

We can see the number of observations in each of the train and test sets for each split match the expectations calculated using the simple arithmetic above.

```
Observations: 1410
Training Observations: 705
Testing Observations: 705
Observations: 2115
Training Observations: 1410
Testing Observations: 705
Observations: 2820
Training Observations: 2115
Testing Observations: 705
```

The plot also shows the 3 splits and the growing number of total observations in each subsequent plot.



Sunspot Dataset Multiple Train-Test Split

Using multiple train-test splits will result in more models being trained, and in turn, a more accurate estimate of the performance of the models on unseen data.

A limitation of the train-test split approach is that the trained models remain fixed as they are evaluated on each evaluation in the test set.

This may not be realistic as models can be retrained as new daily or monthly observations are made available. This concern is addressed in the next section.

Walk Forward Validation

In practice, we very likely will retrain our model as new data becomes available.

This would give the model the best opportunity to make good forecasts at each time step. We can evaluate our machine learning models under this assumption.

There are few decisions to make:

1. **Minimum Number of Observations**. First, we must select the minimum number of observations required to train the model. This may be thought of as the window width if a sliding window is used (see next point).
2. **Sliding or Expanding Window**. Next, we need to decide whether the model will be trained on all data it has available or only on the most recent observations. This determines whether a sliding or expanding window will be used.

After a sensible configuration is chosen for your test-setup, models can be trained and evaluated.

1. Starting at the beginning of the time series, the minimum number of samples in the window is used to train a model.
2. The model makes a prediction for the next time step.
3. The prediction is stored or evaluated against the
4. The window is expanded to include the known va

Because this methodology involves moving along the ed Walk Forward Testing or Walk Forward Validation. Add s used to train a model, this method is also referred to a

This capability is currently not available in scikit-learn, a carefully configured TimeSeriesSplit.

Below is an example of how to split data into train and od.

```python
from pandas import Series
from matplotlib import pyplot
series = Series.from_csv('sunspots.csv', heade
X = series.values
n_train = 500
n_records = len(X)
for i in range(n_train, n_records):
    train, test = X[0:i], X[i:i+1]
    print('train=%d, test=%d' % (len(train), len(test)))
```

Running the example simply prints the size of the training and test sets created. We can see the train set expanding teach time step and the test set fixed at one time step ahead.

Within the loop is where you would train and evaluate your model.

```
train=500, test=1
train=501, test=1
train=502, test=1
train=503, test=1
train=504, test=1
...
train=2815, test=1
train=2816, test=1
train=2817, test=1
train=2818, test=1
train=2819, test=1
```

You can see that many more models are created.

This has the benefit again of providing a much more robust estimation of how the chosen modeling method and parameters will perform in practice. This improved estimate comes at the computational cost of creating so many models.

This is not expensive if the modeling method is simple or dataset is small (as in this example), but could be an issue at scale. In the above case, 2,820 models would be created and evaluated.

As such, careful attention needs to be paid to the window width and window type. These could be adjusted to contrive a test harness on your problem that is significantly less computationally expensive.

Walk-forward validation is the gold standard of model evaluation. It is the k-fold cross validation of the time series world and is recommended for your own projects.

Further Reading

- sklearn.model_selection.TimeSeriesSplit API Doc
- Rolling-Window Analysis of Time-Series Models f
- Backtesting on Wikipedia to learn more about bac

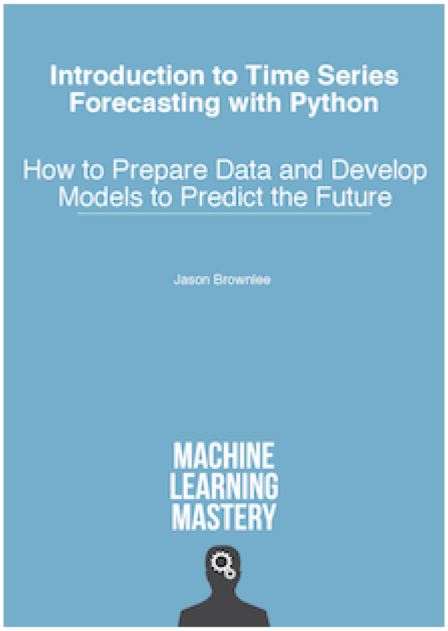
Summary

In this tutorial, you discovered how to backtest machin n.

Specifically, you learned:

- About the importance of evaluating the performar
- How to create train-test splits of time series data, and how to create multiple such splits automatically.
- How to use walk-forward validation to provide the most realistic test harness for evaluating your models.

Do you have any questions about evaluating your time series model or about this tutorial?
Ask your questions in the comments below 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

Tweet Share Share G+

About Jason Brownlee

Jason Brownlee, Ph.D. is a machine learni
with modern machine learning methods via
View all posts by Jason Brownlee →

‹ How To Resample and Interpolate Your Time Series Data With Python

How to Use and Remove Trend Information from Time Series Data in Python ›

93 Responses to *How To Backtest Machine Learning Models for Time Series Forecasting*

Michael December 19, 2016 at 7:57 pm # REPLY

Jason,

second link from “Further Reading” should probably point to mathworks.com instead of amathworks.com, which is not found

Jason Brownlee December 20, 2016 at 7:31

Thanks Michael, fixed!

SalemAmeen December 19, 2016 at 9:32 pm # REPLY

Many thanks, it is short and full of information.

Jason Brownlee December 20, 2016 at 7:30 am # REPLY

I'm glad to hear you found it useful.

Shreyak Tiwari December 20, 2016 at 3:34 am #

For walking forward validation it will consume
and even results won't be much different between eac
each iteration and divide train and test data in that mar
test data could be for h steps for each iteration rather t
my point of view. No hard rules here.

Jason Brownlee December 20, 2016 at 7:27

Hi Shreyak,

Yes, that would be a sampled version of walk-forward validation, a subset.

This is pretty much what the multiple train-test splits provides in the sklearn TimeSeriesSplit object – if I understood you correctly.

Saurabh Bhagvatula December 27, 2016 at 5:18 am # REPLY

My query is related to walk forward validation:

Suppose a time series forecasting model is trained with a set of data and gives a good evaluation with test-set in time_range-1 and model produces a function F1. For time_range-2 and another set of training and testing data model generates function F2. Similarly for time_range-N the model generate Function FN. How the different models when combined and implemented forecast the result based on forecasting function based of local model and not the combined model of all time range model, which may possibly be producing error in forecasting.

Jason Brownlee December 27, 2016 at 5:26 am # REPLY ↩

Hi Saurabh,

Sorry, I don't quite understand the last part of your question. Are you able to restate it?

Ram January 20, 2017 at 12:11 am # REPLY ↩

I am just going through your posts on Time Series. Are you using any particular resource as a reference material for these things ?

Jason Brownlee January 20, 2017 at 10:21 a

A shelf of textbooks mainly 🙂

Ian February 11, 2017 at 10:01 am #

Hi Jason

Thanks so much for this in-depth post. My question is:
Which performance measure should we use in selectin
For example, if I add one test subset at a time in a bina e
either 1 or 0.
In this case, how should I select a model? Should I use other measures instead?

I am building my model as stock price classification where 1 represents up, and 0 means down. I use TimeSeriesSplit and divide into T (sample size) – m (rolling window) + 1.

Thanks a lot and I look forward to listening your insights!

Jason Brownlee February 12, 2017 at 5:33 am # REPLY ↩

Hi Ian,

This is a problem specific question.

Perhaps classification accuracy on the out of sample dataset would be a good way to pick a model in your case?

Ian February 12, 2017 at 9:22 am #

Jason,

Thanks so much for answering.
If we walk one step forward every time just like what you illustrate in the Walk Forward Validation, doesn't that mean the test dataset come from out of sample?

Hope this is not too problem specific, and thanks again in advance.

Jason Brownlee February 13, 2017 at 9:11 am # REPLY

Hi Ian,

Walk forward validation is a method for est
We contrive out of sample and each time s
sample.

We can use the same model in ops, as lor
observation is received.

Does that make sense?

Magnus March 4, 2017 at 2:39 am #

Thanks Jason for an informative post!
If the time series is very long, e.g. minute values for 10
understand you, another way to do this is to fix the length of the training set, e.g. 2 years, but just move it, like this:
Split 1: year 1+2 train, year 3 test
Split 2: year 2+3 train, year 4 test
…
Split 8: year 8+9 train, year 10 test
Is this correct and valid?

Jason Brownlee March 6, 2017 at 10:51 am # REPLY

Sounds good to me.

Also consider how valuable the older data is to fit the model. It is possible data from 10 years ago is not predictive of today, depends on the problem of course.

jessie August 9, 2018 at 5:55 pm # REPLY

well how is this splits code?(8 splits)

jessie August 9, 2018 at 6:35 pm #

REPLY

well ? how will it do ? what is the it's code?

Jason Brownlee August 10, 2018 at 6:11 am #

REPLY

Splitting a dataset is really specific to the dataset.

If you need help getting started with nump
https://machinelearningmastery.com/index
python/

jessiee August 10, 2018 at 8:28 pm #

well which code should we use for thi

Jason Brownlee August 11, 2018 a

Sorry, I cannot write code for you,

marwa May 11, 2017 at 12:14 am #

REPLY

Thank you for your post Jason.

I would like to ask you which model we will chose if we have implementation purpose.
In fact, for example if the time series is hour values of 3 years, walk forward could be applied in this way:

Split 1: year 1 train, year 2 test and we will get model1, error of prediction 1
Split 2: year 1+2 train, year 3 test and we will get model2, error of prediction 2

which model should we then choose ?

Jason Brownlee May 11, 2017 at 8:32 am #

REPLY

Great question.

Pick the model that best represents the performan

Elie Kawerk June 24, 2017 at 8:14 pm # REPLY

Jason,

I think that when Marwa mentioned 'models', she meant applying the same model (such as ARMA) on different data (corresponding to the expanding window).
I think that the walk-forward method, just like k-fold CV, gives an array of metrics whose mean somehow corresponds to the true skill of the model.

I think that when this mean is evaluated, the model should be trained on the entire dataset (check Practical Time Series Forecasting with R- Shmueli) just like with K-fold CV.

Please correct me if I am wrong.

Regards

Jason Brownlee June 25, 2017 at 6

Walk forward validation will give a

Walk forward validation requires some por
evaluate it, and the portion for evaluation i
forward". We do not train on the entire trai
would we compare the prediction to in orde

Shifen June 29, 2017 at 5:46 pm # REPLY

Dear Jason,
Thanks so much for this in-depth post. My question is:
If my time series are discontinuous(such as two weeks in March and two weeks in September), How should I divide the data set?
If I use time series as supervised learning, it could lead to a sample containing data for March and September.
This question has puzzled me for a long time and I look forward to hearing from you.

Jason Brownlee June 30, 2017 at 8:10 am # REPLY

I don't have a good answer.

Perhaps try to fill in the missing time with 0s or nans.
Perhaps try to ignore the missing blocks.
Perhaps focus on building a model at a lower scale

Nick July 7, 2017 at 12:57 pm # REPLY

Hey Jason, can you comment on Rob Hyndman's paper stating that CV can, in fact, be used for time-series data (https://robjhyndman.com/papers/cv-wp.pdf)?

Nick July 7, 2017 at 1:05 pm # REPLY

As a follow-up, would this code work in a time-series context:
http://machinelearningmastery.com/use-keras-deep-learning-models-scikit-learn-python/

Thank you in advance for your guidance!

Jason Brownlee July 9, 2017 at 10:35 a

Try it and see.

Jason Brownlee July 9, 2017 at 10:35 am #

I hope to when I have a pocket of time.

Daniel July 14, 2017 at 7:54 am # REPLY

Is there a way to store the model fit values in such a way that we can update the model after every iteration instead of recreate an entirely new one?
My dataset has 55,000 samples and I want to run a test set of 5,000, but recreating 5,000 models would take roughly 80 hours. Thanks.

Jason Brownlee July 14, 2017 at 8:38 am # REPLY

Yes, here's how to save the model:
http://machinelearningmastery.com/save-arima-time-series-forecasting-model-python/

Daniel July 14, 2017 at 9:38 am # REPLY

Thanks for responding so quickly! Say I trained a model, saved it, ran it on a test sample x1 and then iterated to test the next test sample x2. Once I load the old model, how would I add sample x1 to update the model, potentially making it perform better?
That way I am always predicting sample n+1 with a train set from 0 to n without always creating a new model for the 5000 iterations.

Jason Brownlee July 15, 2017 at 9:34 am # REPLY

You would have to fit the model on just the new data or on a combination of the new and old data.

This can be done with a new model or by u

I do not have an example with ARIMA, but
http://machinelearningmastery.com/update

Huzefa Barwaniwala October 4, 2017 at 3:51 am

Hi, Jason

Thanks a lot for this post, I have recently gone through
and found it quite informative; especially the post on fe
with supervised learning algorithms.

Now, if I have a time series data for demand forecasting, and I have used a lot of feature engineering on the 'date' variable to extract all the seasonality, for example, day, month, the day of the week, if that day was a holiday, quarter, season, etc. I have also used some FE on the target variable to create lag features, min, max, range, average, etc.

My question to you is: Do I still need use backtesting/ Walk Forward Validation? or I can now use a simple k-fold cross validation since the order of time series won't be important?

Thanks a lot. Keep doing this awesome work.

Huzefa

Jason Brownlee October 4, 2017 at 5:50 am # REPLY

It really depends on your data.

In practice, I do recommend walk-forward validation when working with time series data. It is a great way to make sure you are not tricking yourself.

Huzefa Barwaniwala October 5, 2017 at 3:22 am # REPLY

Jason,

Thank you for getting back. Yes, I agree with you. One more thing I realized is, I have made lags as a feature and if in any of the fold of CV a future data is used to predict past then it will act as a target leakage!

Best,
Huzefa

Jason Brownlee October 5, 2017 at 5:28 am REPLY

Absolutely!

Danilo November 4, 2017 at 6:14 am #

Hi Jason

Your posts are really amazing. I have learned a lot rea
me with a doubt regarding backtest and transforming ti

May I used backtest, to identify the best lag for transfo

Jason Brownlee November 5, 2017 at 5:10 am # REPLY

Sure.

annesoj November 4, 2017 at 7:49 pm # REPLY

Hi Jason,

Thank you so much for this post.
However I will have a question that might seems stupid but…

This give me a graphical version of the reality (on the train) and of my predictions (on the test). But it is not an evaluation of my model….

How do I know using those methods, if my models is great or bad?

Imagine I want to try an ARIMA (5,2) and an ARIMA (6,3). How do I do to pick the best one? How do I evaluate each one using “Walk Forward Validation”????

To evaluate the first model, I can do the mean of the error, for each split, between the prediction and the real value?

To pick the best model I can compare those mean between the 2 models?

Would it be a good evaluation methods?

Thank you again!

Jason Brownlee November 5, 2017 at 5:15 am #

REPLY

You can compare the predictions to the expected values and calculate an error score.

Here are some examples:
https://machinelearningmastery.com/time-series-fo

Urfa January 22, 2018 at 10:38 am #

Hi Jason,

I have a set of monthly panel data from 2000 to 2015 a
predict one month ahead by using a (pooled) rolling re
there are better alternatives for panel data like regress
I'm getting accurate predictions.) Regression model lo
+ x10_{i,t} where t is the current month and i is the id.

Furthermore, I select a new model in every step by using a dynamic model selection. In detail:

1. Take a fixed windows size of five years and split it into a training and validation set. The first 58 months as training and the month 59 as validation set.

2. Choose Explanatory Variables or rather a regression model by running a stepwise regression for model selection with the training and validation set and the Average Square Error of the validation set as a criterion.

3. Take the data from month 60 and the regression model from step 2, to make a forecast for month 61.

4. Go to step 1 and roll the window one month forward.

I couldn't find any literature where you select a new regression model or new explanatory variables at every step of the rolling regression. Do you know if there is any literature on that?

Thank you!

Jason Brownlee January 23, 2018 at 7:48 am #

REPLY

Nice!

Good question. Can't think of a good source off the top of my head, I would be sniffing around applied stats books/papers or more likely applied economics works.

Urfa January 23, 2018 at 8:55 am # REPLY

Thank you!

Until now I can't find anything on that approach and I searched several papers and books on that topic. I will keep searching! 🙂

By the way, does this approach makes sense to you?

Jason Brownlee January 24, 2018

Hang in there.

Generally, there is no one size fits all appr

what suits the problem/data/project.

Mohammed Helal February 6, 2018 at 5:57 am #

Correct me if I'm wrong, but it seems to me th

Validation technique, with the exceptions that (1) there is no option for minimum sample size (or a sliding window necessarily), and (2) the predictions are done for a larger horizon.

PS. Thanks a lot for your posts!

Jason Brownlee February 6, 2018 at 9:23 am # REPLY

It is a one-time split, where as walk-forward validation splits on each time step from one point until the end of the dataset.

Does that help?

Alexis March 2, 2018 at 5:14 pm # REPLY

Hi Jason, I don't see why TimeSeriesSplit makes such a "complicated" formula to create a test set of constant size. I would rather make it as a proportion of the whole window at the first iteration, and then keep that length for the remaining steps. Would it be correct ?

Jason Brownlee March 3, 2018 at 8:06 am # REPLY

Yes, nice one. You have essentially described a variation on walk forward validation.

Tarun March 15, 2018 at 10:31 pm # REPLY

Hi Jason,

I have a query regarding Walk forward validation of TS. Let's say I need to forecast for next 3 months (Jan-Mar 18) using last 5 years of data (Jan13-Dec 17).
In principle I would want to use Walk forward as I woul
unseen data. I'd use your approach which is:

1) Set Min no of observations : Jan12-Dec 16
2) Expanding TEST window : Entire 2017, which mear
iteration 1 and in next iteration, Jan 17 becomes part o
entire 2017.

My question is why do I need to RETRAIN model ever
next 3 TEST points assuming the fact that model that I

Can't I select (let's say) top 5 models from step 1,calcu
window) and select the one with least RMSE?.

Eagerly awaiting your reply!

Jason Brownlee March 16, 2018 at 6:19 am # REPLY

You can, but the further you get from real obs as inputs the worse model skill will be come.

This post will give you some additional ideas on performing a multi-step forecast without using obs as inputs:
https://machinelearningmastery.com/multi-step-time-series-forecasting/

Tarun March 16, 2018 at 10:55 pm # REPLY

Hi Jason,

Thanks for the reply.

"the further you get from real obs" by that do you mean to say that I am not retraining my model using real data?

Jason Brownlee March 17, 2018 at 8:37 am # REPLY

I mean that the longer the lead time – further you forecast into the future, the less stable/skillful the results.

Tarun March 19, 2018 at 4:44 pm #

Thanks Jason. You are indeed doing a great job.

Jason Brownlee March 20, 20

Thanks.

Ha Pham April 17, 2018 at 1:04 pm #

Hi Jason,

Thanks a lot for your post. I am working on a demand
only have sales data of two years. Unit of data point ca
60% of the products have lots of zero and some bursty
through out the years. I tried two approaches:
– Using sales data of previous 4 weeks to train and predict sales of next week
– Using sales data of year 1 to predict the whole sales data of next year with no update to the model

My questions:
– Is there any theoretical error in these approaches? I can clarify a few things more if you need
– In this post you only talk about one time series. Can this be applied to my case where I have thousands of time series needed to be forecast at the same time?
– For this kind of problem, which algorithm tend to give best result? Can an out-of-the-box algo like XGBoost do the job? I have browsed through some papers and they introduced different methods like Neural Networks or Bayesian methods, which haven't touched yet.

Thanks.

Jason Brownlee April 17, 2018 at 2:54 pm # REPLY

That sounds like a great problem.

I'm eager to dive in and offer some advice, but it would be a big time investment for me, I just don't have the capacity. I hope to have more posts on this top

In general, I recommend testing a suite of modeling methods and framings of a problem to help you discover what works best for your specific dataset.

I'm eager to hear how you go.

Dicky Chou April 23, 2018 at 6:26 pm # REPLY

Hi Jason
I am a meteorologist currently working on a time series verification problem.
My colleagues make forecasts every day and I hope to evaluate the accuracy of them.
I find that there are some time shift between our forecast and the observation. For example, we think it will be raining at 5 am tomorrow. However, the rain happer
such as contingent table, we get a miss and a false ala
inappropriate in this case since we the weather conditi
temporal attribution of these data. Can you give me so
series data?

Jason Brownlee April 24, 2018 at 6:25 am #

Great question.

I believe there is a huge body of literature on weat
area).

Here is a great place to start:
http://www.cawcr.gov.au/projects/verification/

Deniz May 16, 2018 at 10:09 am # REPLY

Hi Jason,

Is using AIC for forecasting a good method? Or should I use cross-validation while building forecasting models?

Jason Brownlee May 17, 2018 at 6:21 am # REPLY

It really depends on your project goals and on your specific data.

Mustafa Qamar-ud-Din June 16, 2018 at 7:10 p

Thank you for informative series. I would probably have to read it again, but if you could please correct me whether Sliding Window and Backtest mean the same thing. In the sense that you move the window forward step at a time?

Jason Brownlee June 17, 2018 at 5:39 am #

REPLY

Sliding window refers to a way of framing a time series as a supervised learning problem.

Backtesting is a general term for evaluating a model on historical data.

Anthony The Koala June 20, 2018 at 10:38 am #

Dear Dr Jason,
For those having difficulty plotting the data sourced fro
https://datamarket.com/data/set/22ti/zuerich-monthly-s
helpful before even using Python.

Even if you imported the file from the website as a CS
extraneous information at the bottom of the spreadshe
not cleaned, Python will produce error messages.
.
(1) Open the sunspot.csv file into a spreadsheet progra
(2) Leave the header at the top of the file alone.
(3) Scroll down to the very end of the data file (2821 rows down). Delete the rows containing NaN and text
"Zuerich monthly sunspot numbers 1749-1983".
(4) Save the file as sunspot.csv in CSV format
(3) In Python import the data as usual

```
Series.read_csv('sunspot.csv', header=0)
```

Everything should be OK from that point.

Thank you,
Anthony of Sydney

Jason Brownlee June 21, 2018 at 6:03 am #

REPLY

I also provode all datasets in CSV format here:
https://github.com/jbrownlee/Datasets

Gautam July 2, 2018 at 1:22 am #

REPLY

Hello Jason,

You have become a one stop website for machine learning. Thank you for all the efforts!
I am little stuck and validate my approach here, if you can:
I am trying to predict a stock market index using multiple time series: ex say many commodity indexes besides the targeted index itself. Is this approach terribly wrong? If not can you please possible point to good start point. I am really stuck here badly. Appreciate your thoughts

Jason Brownlee July 2, 2018 at 6:25 am # REPLY ↩

Thanks.

This is a common question that I answer here:
https://machinelearningmastery.com/faq/single-faq
or-the-stock-market

Gautam July 2, 2018 at 1:25 am #

Just additional comment to my previous comr
problem using supervised ml method such as Random

Kingsley Udeh July 12, 2018 at 7:39 pm #

Hi Jason,

Thanks as always.

Please how do I train and evaluate my model within the loop of a Walk Forward Validation approach?

Within the Walk Forward Validation, after choosing my min training size, I created, say for, eg.

```
range of train to len(records):
train, test = X_obs[0:i], X_obs[i:i+1]
# Fit Model
history = model.fit(train_X, train_y, epochs=1000, batch_size=4192, validation_data= (test_X, test_y),
verbose=0, shuffle=False)
# Evaluate Model
loss = model.evaluate(test_X, test_y, verbose=0)
print(loss)
model.save('lstm_model.h5')
```

At the end, I have 10 different loss or validation scores. Is the last saved model the average of all the 10 models? How do I make predictions and calculate the RMSE for the average model?

I'm still learning the Walk Forward Validation method a
thing to do.

I look forward to hearing from you soon .

Jason Brownlee July 13, 2018 at 7:38 am # REPLY

I recommend not using a validation set when fitting the model. The skill score is calcualted based on the predictions made within the walk-forward validation loop.

Kingsley Udeh August 3, 2018 at 7:15 am # REPLY

I used validation set because I wante
modelcheckpoint. Thus, I would pick the best r
independent test set.

In addition, I would use the method or approac
model and compare the final model with the m

Abdessamad July 30, 2018 at 7:42 pm #

Hi Jason,

Thanks a lot for your post. You said in the Walk Forwa
models would be created and evaluated."" Is it not 2,32
minimum ?

Thanks.

Jason Brownlee July 31, 2018 at 6:00 am # REPLY

Yes. Nice catch.

Prince Grover August 10, 2018 at 8:15 am # REPLY

Hi Jason,

Thanks for the article. I like the walk forward validation approach. I am currently using the same approach in one of the problem and have a question that I would like to discuss with you.

Q: How can we make train, validation and test split using walk forward validation approach? We generally split data into 3 parts and keep a separate test data for final evaluation. If we are keeping a window of width w and sliding it over next days, I can use to either tune hyperparameters or final validation score. What about test score and generalizability of our model?

Thanks in advance!

Jason Brownlee August 10, 2018 at 2:16 pm # REPLY

Good question.

Perhaps choose a period over which the model will make predictions, not be updated with true values and the holdout set can be used as validation for tuning the model?

Philip P August 18, 2018 at 12:55 am #

Jason,

So, I'm wondering how these folds from Walk Forward
as a CV object into a sklearn model like xgboost. I've u
folds before. My project at work has sales data for a nu
will predict sales 4 weeks out by each store. Right now
with a minimum train size of 52 weeks and test on the
Here, n_train = 52 and max_week = 80. My code and c

Code:

```
for i in range(n_train, max_week):
if i % 4 == 0:
train, test = df[(df.WeekCount_ID >= 1) & (df.WeekCount_ID i) & (df.WeekCount_ID <= i + 4)]
print('train=%d, test=%d' % (len(train), len(test)))
```

Output:

```
train=3155, test=260
train=3415, test=260
train=3675, test=260
train=3935, test=260
train=4195, test=272
train=4467, test=282
train=4749, test=287
```

Jason Brownlee August 18, 2018 at 5:40 am # REPLY

Good question.

I write my own validation and grid search procedures for time series, it's also my general recommendation in order to give more control. The

Philip P. August 21, 2018 at 12:56 am # REPLY

Jason, thanks for the quick reply. So, for someone who is learning all of this concurrently (machine learning, time series, python, sql, etc) and not sure how to write my own python procedures, is this custom code of yours something that you cover in any of your books? If not, is this something that you would share or that I could find posted on another forum? Thanks again.

Jason Brownlee August 21, 2018 at 6:18 am # REPLY

I give an example of custom code

provide.

I have a new book coming out in a week o

Dimos September 20, 2018 at 4:30 pm #

Hi Jason,

Amazing tutorial!!!!!!!!

Let's assume that i have training data for periods 1-10(
Should i predict the target variable for period 101 and t

Many thanks

Jason Brownlee September 21, 2018 at 6:24 am # REPLY

Yes, you can use the real observation is available or use the prediction as the input. The latter is called a recursive model:
https://machinelearningmastery.com/multi-step-time-series-forecasting/

Tianyu September 24, 2018 at 8:51 pm # REPLY

Hi Jason,
May I ask two questions?
1. How to apply early stopping in walk forward validation to select the model in each walk forward step?
2. I think for time series data, we can Convert a Time Series to a Supervised Learning Problem. As the result, each sample is consist of past time step data as input and one target output. Every sample is now

independent and there is no time order existed when using stateless LSTM for training. We can now shuffle all the samples and split the data as training and validation set as normal. Correct me if I am wrong.

Jason Brownlee September 25, 2018 at 6:20 am # REPLY

You can do early stopping on the fitting of the model prior to making a prediction in each step of the validation process.

Perhaps. It might depend on the specifics of your domain.

Tianyu September 25, 2018 at 8:05 am #

Thanks for your reply.
If the model is to predict classification problem
cannot be used for validation based early stop

Jason Brownlee September 25, 20

Why not?

Tianyu September 26, 2018 at 6:18 am #

Do you mean we can make it like if for 10 epochs' accuracy is 1 then stop training? But in this situation how to compare two models in two epochs with same accuracy=1? I mean if there are many samples for validation, I can save the best model with highest val_acc by check point function from Keras.

Jason Brownlee September 26, 2018 at 2:21 pm #

Not sure I follow.

Early stopping with time series is hard, but I think it is possible (happy to be proven wrong). Careful attention will need to be paid to exactly what samples are used as the validation set each step.

I don't have a worked example, sorry.

Venkata phanikrishna September 30, 2018 at 9:54 pm # REPLY

Hi Jason,
I am new to the ML. I understood ML topics theoretically. Coming to the implementation case, really it is very hard for me. Through your website, I did some implementation work. Thanks for your help.

Coming to my question,
how to use ML binary classification concepts in case of nonstationary data (Example: EEG data)?

At present, with the help of available samples, I train the model using KV fold cross-validation.

clf=ML-Classifcationmodel();
y_pred = cross_val_predict(clf,MyX,MyY,cv=10)

every time I am getting the same results.

but if I shuffle the samples before training using below
from sklearn.utils import shuffle
mydataset = shuffle(df1)

how to find the best model in such cases.

Jason Brownlee October 1, 2018 at 6:25 am

It's not valid to use cross validation for tim

The train/test data must be split in such a way as t
never trained on data from the future and only tested on data from the future.

Yue Lee October 25, 2018 at 2:15 am # REPLY

In this post, it is explained that a Time Series problem could be reframed as a machine learning one with inputs and outputs. Could we consider in this case that each row is an independent observation and use Cross Validation , Nested Cross validation or any method for hyperparameters tuning and validation?

Jason Brownlee October 25, 2018 at 8:03 am # REPLY

Nearly. The problem is that rows that contain information about the future in the training set will bias the model.

Leave a Reply

Name (required)

Email (will not be published) (requ

Website

SUBMIT COMMENT

Welcome to Machine Learning Mastery!

Hi, I'm Jason Brownlee, Ph.D.
My goal is to make developers like *YOU*

Read More

Time Series Forecasting with Python
Explore time series data and predict the future.

Introduction to Time Series Forecasting with Python

How to Prepare Data and Develop Models to Predict the Future

Jason Brownlee

MACHINE LEAR MAS

POPULAR

How to Develop a Deep Learning Photo Caption Ge
NOVEMBER 27, 2017

How to Develop a Neural Machine Translation Syst
JANUARY 10, 2018

How to Develop an Encoder-Decoder Model for Sequence-to-Sequence Prediction in Keras
NOVEMBER 2, 2017

Difference Between Classification and Regression in Machine Learning
DECEMBER 11, 2017

How to Develop a Word-Level Neural Language Model and Use it to Generate Text
NOVEMBER 10, 2017

How to Develop an N-gram Multichannel Convolutional Neural Network for Sentiment Analysis
JANUARY 12, 2018

So, You are Working on a Machine Learning Problem…
APRIL 4, 2018

You might also like…

- How to Install Python for Machine Learning

- Your First Machine Learning Project in Python
- Your First Neural Network in Python
- Your First Classifier in Weka
- Your First Time Series Forecasting Project

Privacy | Disclaimer | Terms | Contact