Sensitivity Analysis of History Size to Forecast Skill with ARIMA in Python

by **Jason Brownlee** on March 27, 2017 in **Time Series**

How much history is required for a time series forecast model?

This is a problem-specific question that we can investigate by designing an experiment.

In this tutorial, you will discover the effect that history size has on the skill of an ARIMA forecast model in Python.

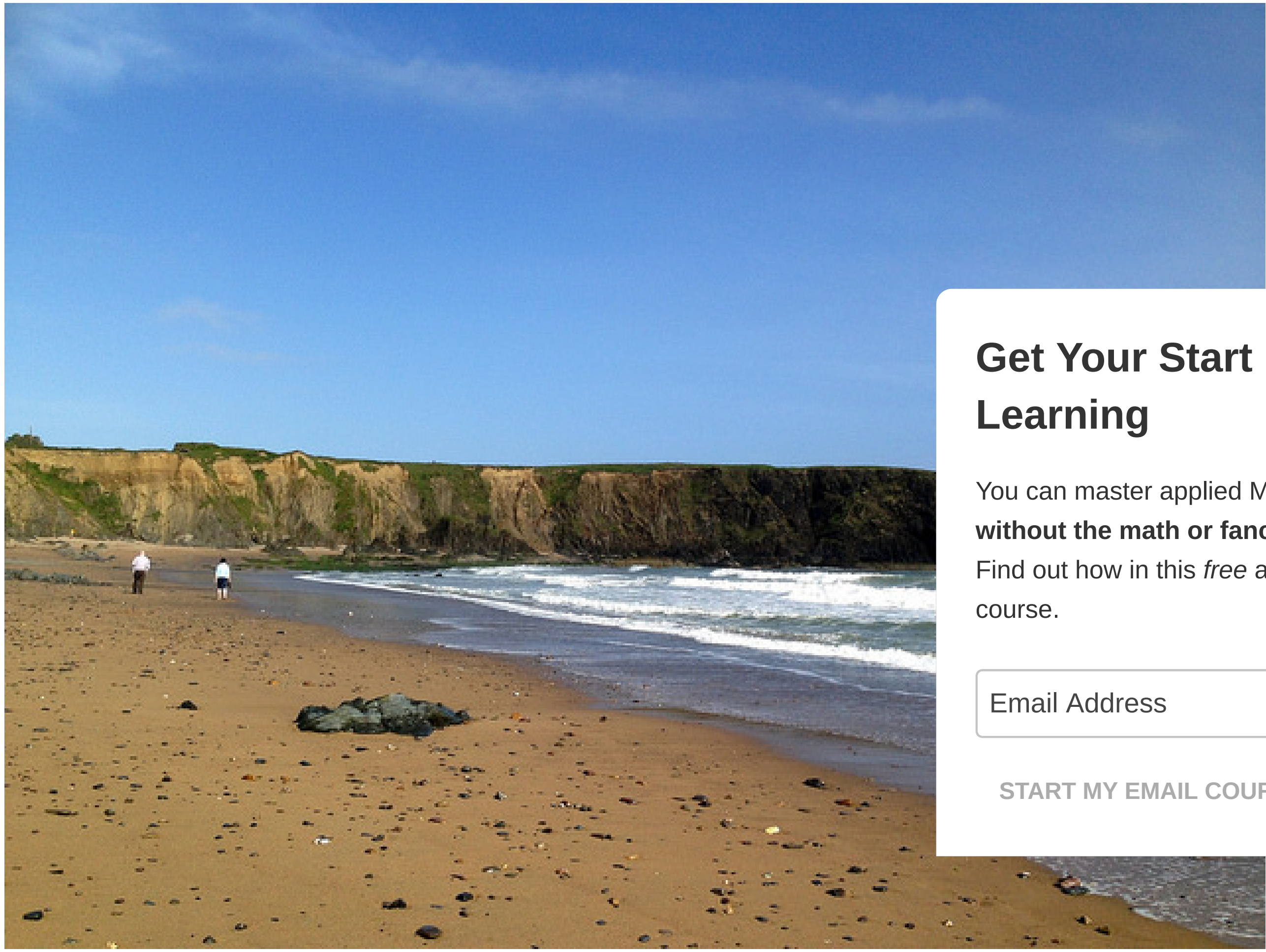
Specifically, in this tutorial, you will:

- Load a standard dataset and fit an ARIMA model.
- Design and execute a sensitivity analysis of the number of years of historic data to model skill.
- Analyze the results of the sensitivity analysis.

This will provide a template for performing a similar sensitivity analysis of historical data set size on y

Let’s get started.

- **Update Aug/2017**: Fixed a bug where the models were constructed on the raw data instead of the seasonally differenced version of the data. Thanks David Ravnsborg!



Sensitivity Analysis of History Size to Forecast Skill with ARIMA in Python
Photo by Sean MacEntee, some rights reserved.

Minimum Daily Temperatures Dataset

This dataset describes the minimum daily temperatures over 10 years (1981-1990) in the city of Mel

The units are in degrees Celsius and there are 3,650 observations. The source of the data is credite[redacted]

Download the dataset and save it in your current working directory with the filename "*daily-minimum-temperatures.csv*".

Note: The downloaded file contains some question mark ("?") characters that must be removed before you can use the dataset. Open the file in a text editor and remove the "?" characters. Also, remove any footer information in the file.

The example below loads the dataset as a Pandas Series.

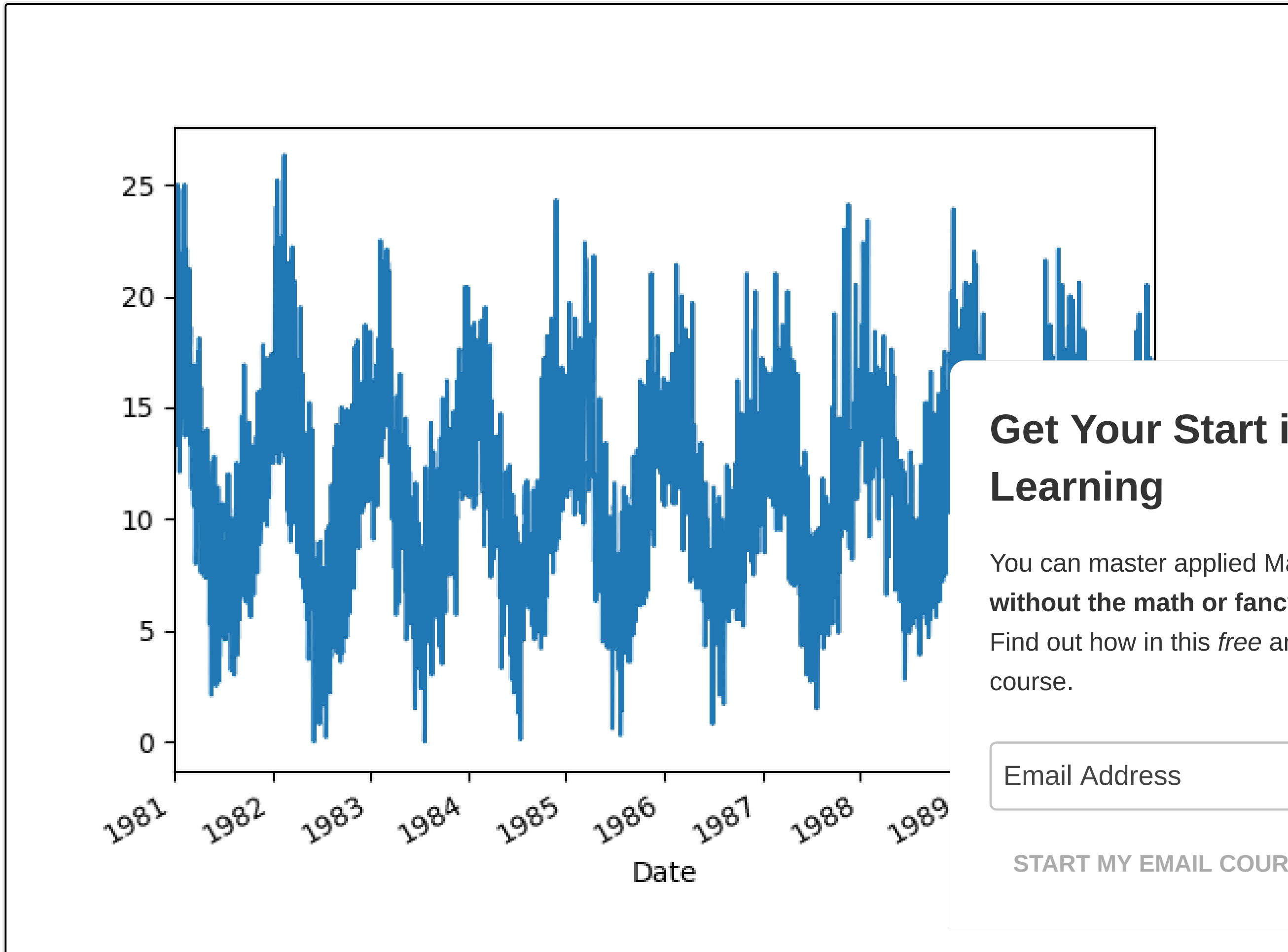
```
# line plot of time series
from pandas import Series
from matplotlib import pyplot
# load dataset
series = Series.from_csv('daily-minimum-temperatures.csv', header=0)
# display first few rows
print(series.head(20))
# line plot of dataset
series.plot()
pyplot.show()
```

Running the example prints the first 20 rows of the loaded file.

```
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
1981-01-06 15.8
1981-01-07 15.8
1981-01-08 17.4
1981-01-09 21.8
1981-01-10 20.0
1981-01-11 16.2
1981-01-12 13.3
1981-01-13 16.7
1981-01-14 21.5
1981-01-15 25.0
1981-01-16 20.7
1981-01-17 20.6
1981-01-18 24.8
1981-01-19 17.7
1981-01-20 15.5
```

```
22 Name: Temp, dtype: float64
```

Then the data is graphed as a line plot showing the seasonal pattern.



Minimum Daily Temperatures Dataset Line Plot

ARIMA Forecast Model

In this section, we will fit an ARIMA forecast model to the data.

The parameters of the model will not be tuned, but will be skillful.

The data contains a one-year seasonal component that must be removed to make the data stationar[redacted]

We can take the seasonal difference by subtracting the observation from one year ago (365 days). This is rough in that it does not account for leap years. It also means that the first year of data will be unavailable for modeling as there is no data one year before to difference the data.

```
# seasonal difference
differenced = series.diff(365)
# trim off the first year of empty data
differenced = differenced[365:]
```

We will fit an ARIMA(7,0,0) model to the data and print the summary information. This demonstrates that the model is stable.

```
# fit model
model = ARIMA(differenced, order=(7,0,0))
model_fit = model.fit(trend='nc', disp=0)
print(model_fit.summary())
```

Putting this all together, the complete example is listed below.

```
# fit an ARIMA model
from pandas import Series
from matplotlib import pyplot
from statsmodels.tsa.arima_model import ARIMA
# load dataset
series = Series.from_csv('daily-minimum-temperatures.csv', header=0)
# seasonal difference
differenced = series.diff(365)
# trim off the first year of empty data
differenced = series[365:]
# fit model
model = ARIMA(differenced, order=(7,0,0))
model_fit = model.fit(trend='nc', disp=0)
print(model_fit.summary())
```

Running the example provides a summary of the fit ARIMA model.

```
                              ARMA Model Results
==============================================================================
Dep. Variable:                   Temp   No. Observations:                 3285
Model:                     ARMA(7, 0)   Log Likelihood               -8690.089
Method:                       css-mle   S.D. of innovations              3.409
Date:                Fri, 25 Aug 2017   AIC                          17396.178
Time:                        15:02:59   BIC                          17444.955
Sample:                    01-01-1982   HQIC                         17413.643
```

```
                    - 12-31-1990
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1.Temp     0.5278      0.017     30.264      0.000       0.494       0.562
ar.L2.Temp    -0.1099      0.020     -5.576      0.000      -0.149      -0.071
ar.L3.Temp     0.0286      0.020      1.441      0.150      -0.010       0.067
ar.L4.Temp     0.0307      0.020      1.549      0.122      -0.008       0.070
ar.L5.Temp     0.0090      0.020      0.456      0.648      -0.030       0.048
ar.L6.Temp     0.0164      0.020      0.830      0.407      -0.022       0.055
ar.L7.Temp     0.0272      0.017      1.557      0.120      -0.007       0.061
                                    Roots
=============================================================================
                 Real           Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           1.3305           -0.0000j            1.3305           -0.0000
AR.2           0.9936           -1.1966j            1.5553           -0.1397
AR.3           0.9936           +1.1966j            1.5553            0.1397
AR.4          -0.2067           -1.7061j            1.7186           -0.2692
AR.5          -0.2067           +1.7061j            1.7186            0.2692
AR.6          -1.7536           -0.8938j            1.9683           -0.4250
AR.7          -1.7536           +0.8938j            1.9683            0.4250
-----------------------------------------------------------------------------
```

Model History Sensitivity Analysis

In this section, we will explore the effect that history size has on the skill of the fit model.

The original data has 10 years of data. Seasonal differencing leaves us with 9 years of data. We will perform walk-forward validation across this final year. The day-by-day forecasts will be collected and

The day-by-day forecasts will be collected and a root mean squared error (RMSE) score will be calc

The snippet below separates the seasonally adjusted data into training and test datasets.

```
train, test = differenced[differenced.index < '1990'], differenced['1990']
```

It is important to choose an interval that makes sense for your own forecast problem.

We will evaluate the skill of the model with the previous 1 year of data, then 2 years, all the way back through the 8 available years of historical data

A year is a good interval to test for this dataset given the seasonal nature of the data, but other inter[redacted] year intervals.

The snippet below shows how we can step backwards by year and cumulatively select all available observations.

For example

- Test 1: All data in 1989
- Test 2: All data in 1988 to 1989

And so on.

```
# split
train, test = differenced[differenced.index < '1990'], differenced['1990']
years = ['1989', '1988', '1987', '1986', '1985', '1984', '1983', '1982']
for year in years:
    # select data from 'year' cumulative to 1989
    dataset = train[train.index >= year]
```

The next step is to evaluate an ARIMA model.

We will use walk-forward validation. This means that a model will be constructed on the selected his
1990). The real observation for that time step will be added to the history, a new model constructed,

The forecasts will be collected together and compared to the final year of observations to give an err
scores and will be in the same scale as the observations themselves.


```
# walk forward over time steps in test
values = dataset.values
history = [values[i] for i in range(len(values))]
predictions = list()
test_values = test.values
for t in range(len(test_values)):
    # fit model
    model = ARIMA(history, order=(7,0,0))
    model_fit = model.fit(trend='nc', disp=0)
    # make prediction
    yhat = model_fit.forecast()[0]
    predictions.append(yhat)
    history.append(test_values[t])
```

```
rmse = sqrt(mean_squared_error(test_values, predictions))
print('%s-%s (%d values) RMSE: %.3f' % (years[0], year, len(values), rmse))
```

Putting this all together, the complete example is listed below.

```
# fit an ARIMA model
from pandas import Series
from matplotlib import pyplot
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error
from math import sqrt
# load dataset
series = Series.from_csv('daily-minimum-temperatures.csv', header=0)
# seasonal difference
differenced = series.diff(365)
# trim off the first year of empty data
differenced = differenced[365:]
# split
train, test = differenced[differenced.index < '1990'], differenced['1990']
years = ['1989', '1988', '1987', '1986', '1985', '1984', '1983', '1982']
for year in years:
    # select data from 'year' cumulative to 1989
    dataset = train[train.index >= year]
    # walk forward over time steps in test
    values = dataset.values
    history = [values[i] for i in range(len(values))]
    predictions = list()
    test_values = test.values
    for t in range(len(test_values)):
        # fit model
        model = ARIMA(history, order=(7,0,0))
        model_fit = model.fit(trend='nc', disp=0)
        # make prediction
        yhat = model_fit.forecast()[0]
        predictions.append(yhat)
        history.append(test_values[t])
    rmse = sqrt(mean_squared_error(test_values, predictions))
    print('%s-%s (%d values) RMSE: %.3f' % (years[0], year, len(values), rmse))
```


Running the example prints the interval of history, number of observations in the history, and the RMSE skill of the model trained with that history.

The example does take awhile to run as 365 ARIMA models are created for each cumulative interval of historic training data.

```
1989-1989 (365 values) RMSE: 3.120
1989-1988 (730 values) RMSE: 3.109
```

```
1989-1987 (1095 values) RMSE: 3.104
1989-1986 (1460 values) RMSE: 3.108
1989-1985 (1825 values) RMSE: 3.107
1989-1984 (2190 values) RMSE: 3.103
1989-1983 (2555 values) RMSE: 3.099
1989-1982 (2920 values) RMSE: 3.096
```

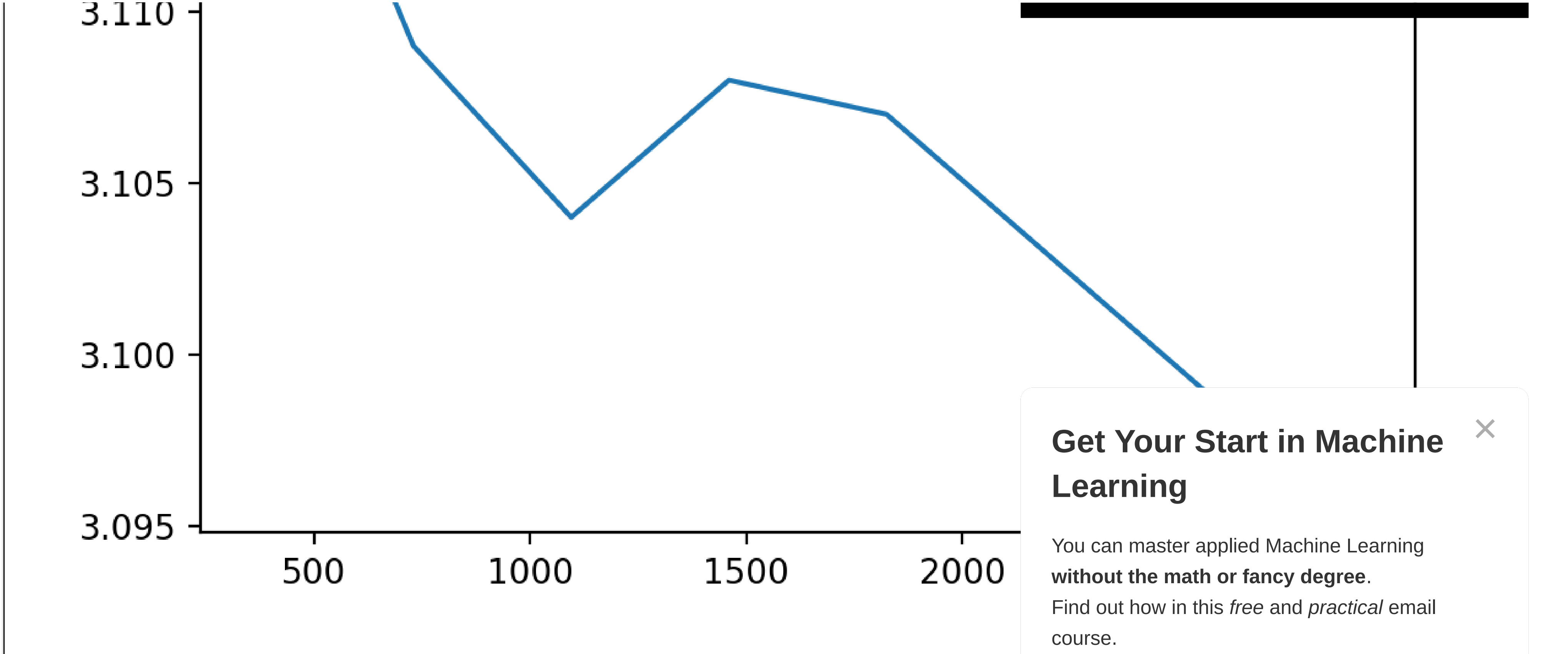
The results show that as the size of the available history is increased, there is a decrease in model error, but the trend is not purely linear.

We do see that there may be a point of diminishing returns at 2-3 years. Knowing that you can use fewer years of data is useful in domains where data availability or long model training time is an issue.

We can plot the relationship between ARIMA model error and the number of training observations.

```
from matplotlib import pyplot
x = [365, 730, 1095, 1460, 1825, 2190, 2555, 2920]
y = [3.120, 3.109, 3.104, 3.108, 3.107, 3.103, 3.099, 3.096]
pyplot.plot(x, y)
pyplot.show()
```

Running the example creates a plot that almost shows a linear trend down in error as training sampl



History Size vs ARIMA Model Error

This is generally expected, as more historical data means that the coefficients may be better optimiz om more years of data, for the most part.

There is also a counter-intuition. One may expect the performance of the model to increase with more history, as the data from the most recent years may be more like the data next year. This intuition is perhaps more valid in domains subjected to greater concept drift.

Extensions

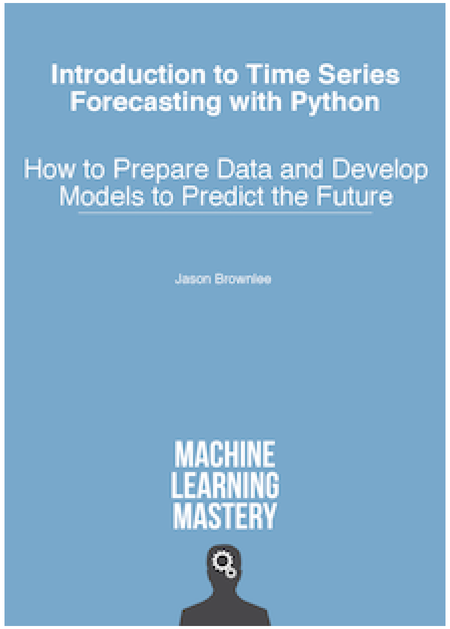
This section discusses limitations and extensions to the sensitivity analysis.

- **Untuned Model**. The ARIMA model used in the example is by no means tuned to the problem. [redacted] history would be performed with an already tuned ARIMA model or a model tuned to each case.
- **Statistical Significance**. It is not clear whether the difference in model skill is statistically significant. Pairwise statistical significance tests can be used to tease out whether differences in RMSE are meaningful.
- **Alternate Models**. The ARIMA uses historical data to fit coefficients. Other models may use the increasing historical data in other ways. Alternate nonlinear machine learning models may be investigated.
- **Alternate Intervals**. A year was chosen to joint the historical data, but other intervals may be used. A good interval might be weeks or months within one or two years of historical data for this dataset, as the extreme recency may bias the coefficients in useful ways.

Summary

In this tutorial, you discovered how you can design, execute, and analyze a sensitivity analysis of the
model.

Do you have any questions?
Ask your questions in the comments and I'll do my best to answer.



Click to learn more.

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 Make Out-of-Sample Forecasts with ARIMA in Python

Feature Selection for Time Series Forecasting with Python ›

Get Your Start in Machine Learning

8 Responses to *Sensitivity Analysis of History Size to Forecast Skill with ARIMA in Python*

sura April 7, 2017 at 3:39 am # REPLY

thank you ! but, can you give me a download address of dataset? i want to try again!
thank!

Jason Brownlee April 9, 2017 at 2:46 pm # REPLY

Sorry, here it is:
https://datamarket.com/data/set/2324/daily-minimum-temperatures-in-melbourne-australia-1981-199

Eugeniy May 5, 2017 at 6:44 pm #

Good afternoon!
Thank you for your article.
Tell me please, if there are more formal and mathematical definitions of sensitivity analysis of history size
Or is this usually only an experimental way of determining?
Thank you!

Jason Brownlee May 6, 2017 at 7:40 am # REPLY

I'm sure you can analyze the effect of history size on the model analytically.

A sensitivity analysis seeks to answer the question empirically.

David Ravnsborg August 18, 2017 at 9:36 am # REPLY

Why do you declare "differenced" and then immediately write over it without using it?

Jason Brownlee August 18, 2017 at 4:37 pm # REPLY

Good question, that looks like a bug to me. I'll add a note to trello to fix it up.

David Ravnsborg August 18, 2017 at 5:48 pm #

I looked into it a little further on my end. I think it was just a typo where:
# seasonal difference
differenced = series.diff(365)
# trim off the first year of empty data
differenced = series[365:]

should have been…

# seasonal difference
differenced = series.diff(365)
# trim off the first year of empty data
differenced = differenced[365:]

But it completely changes the results for the worse 🙁 Any chance you could cover this? It would make a great follow-up. Here are the results I get:
model.py:496: ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check mle_retvals
"Check mle_retvals", ConvergenceWarning)
1989-1989 (365 values) RMSE: 3.120
1989-1988 (730 values) RMSE: 3.109
model.py:496: ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check mle_retvals
"Check mle_retvals", ConvergenceWarning)
1989-1987 (1095 values) RMSE: 3.104

1989-1986 (1460 values) RMSE: 3.108
1989-1985 (1825 values) RMSE: 3.107
model.py:496: ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check mle_retvals
"Check mle_retvals", ConvergenceWarning)
1989-1984 (2190 values) RMSE: 3.103
1989-1983 (2555 values) RMSE: 3.099
1989-1982 (2920 values) RMSE: 3.096

Jason Brownlee August 25, 2017 at 3:15 pm #

REPLY

I have updated the post, thanks again David.

We still see the same linearly downward trend in error.

Remember that the RMSE scores are in fact in the units of seasonally differenced temperatu

If you're interested in better results, you can try using a grid search on the ARIMA paramete whether performing a seasonal difference results in better final RMSE on this problem.

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

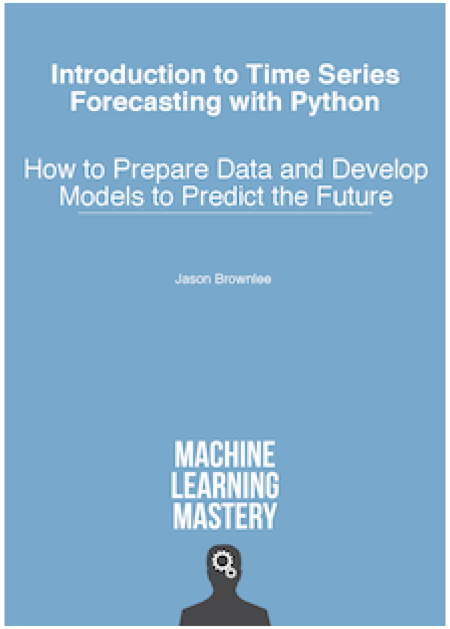
SUBMIT COMMENT

Welcome to Machine Learning Mastery

Hi, I'm Dr. Jason Brownlee.
My goal is to make practitioners like YOU awesome at applied machine learning.

Read More

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

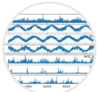
Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras
JULY 26, 2016

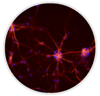
How to Setup a Python Environment for Machine Learning and Deep Learning with Anaconda
MARCH 13, 2017


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

Privacy | Contact | About