Introduction to Time Series Forecasting with Python

7-Day Mini-Course

Jason Brownlee

MACHINE LEARNING MASTERY

Disclaimer

Introduction to Time Series Forecasting with Python

Edition: v1.5

Find the latest version of this guide online at: http://MachineLearningMastery.com

Contents

Before We Get Started...

Python is one of the fastest-growing platforms for applied machine learning. In this mini-course, you will discover how you can get started, build accurate models and confidently complete predictive modeling time series forecasting projects using Python in 7 days. Let's get started.

Who Is This Mini-Course For?

Before we get started, let's make sure you are in the right place. The list below provides some general guidelines as to who this course was designed for. Don't panic if you don't match these points exactly, you might just need to brush up in one area or another to keep up.

- **You're a Developer**: This is a course for developers. You are a developer of some sort. You know how to read and write code. You know how to develop and debug a program.
- **You know Python**: This is a course for Python people. You know the Python programming language, or you're a skilled enough developer that you can pick it up as you go along.
- **You know some Machine Learning**: This is a course for novice machine learning practitioners. You know some basic practical machine learning, or you can figure it out quickly.

This mini-course is neither a textbook on Python or a textbook on time series forecasting. It will take you from a developer that knows a little machine learning to a developer who can get time series forecasting results using the Python ecosystem, the rising platform for professional machine learning.

Note: This mini-course assumes you have a working Python 2 or 3 SciPy environment with at least NumPy, Pandas, scikit-learn and Statsmodels installed.

Mini-Course Overview (what to expect)

This mini-course is broken down into 7 lessons. You could complete one lesson per day (recommended) or complete all of the lessons in one day (hardcore). It really depends on the time you have available and your level of enthusiasm.

Below are 7 lessons that will get you started and productive with machine learning in Python:

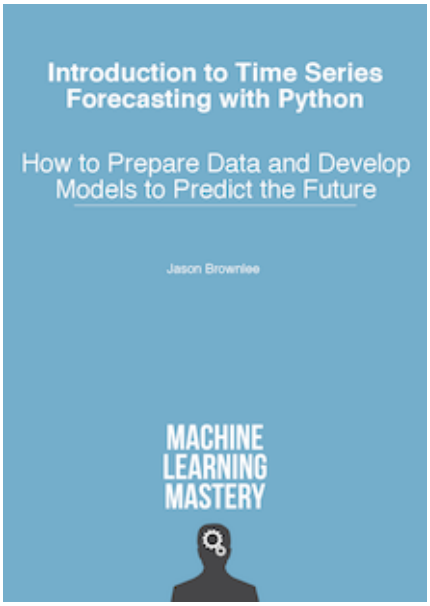
- **Lesson 01**: Time Series as Supervised Learning.
- **Lesson 02**: Load Time Series Data.
- **Lesson 03**: Data Visualization.
- **Lesson 04**: Persistence Forecast Model.
- **Lesson 05**: Autoregressive Forecast Model.
- **Lesson 06**: ARIMA Forecast Model.
- **Lesson 07**: Hello World End-to-End Project.

Each lesson could take you 60 seconds or up to 30 minutes. Take your time and complete the lessons at your own pace. The lessons expect you to go off and find out how to do things. I will give you hints, but part of the point of each lesson is to force you to learn where to go to look for help on and about the Python platform for time series.

Here's a tip: All of the answers these lessons can be found on this blog http://MachineLearningMastery.com. Use the search feature.

Hang in there, don't give up!

If you would like me to step you through each lesson in great detail (and much more), take a look at my book: **Introduction to Time Series Forecasting with Python:**



Learn more here:

- Introduction to Time Series Forecasting with Python

Lesson 01: Time Series as Supervised Learning

Time series problems are different to traditional prediction problems. The addition of time adds an order to observations that both must be preserved and can provide additional information for learning algorithms. A time series dataset may look like the following:

```
Time, Observation
day1, obs1
day2, obs2
day3, obs3
```

Listing 1: Example of a time series.

We can reframe this data as a supervised learning problem with inputs and outputs to be predicted. For example:

```
Input, Output
?,     obs1
obs1,  obs2
obs2,  obs3
obs3,  ?
```

Listing 2: Example of a time series converted to supervised learning.

You can see that the reframing means we have to discard some rows with missing data. Once it is reframed, we can then apply all of our favorite learning algorithms like k-Nearest Neighbors and Random Forest. For more help, see the post:

- Time Series Forecasting as Supervised Learning

In the next lesson, you will discover how to load and explore time series data.

Lesson 02: Load Time Series Data

Before you can develop forecast models, you must load and work with your time series data. Pandas provides tools to load data in CSV format. In this lesson, you will download a standard time series dataset, load it in Pandas and explore it.

Download the daily female births dataset from DataMarket in CSV format and save it with the filename `daily-births.csv`.

- Download Dataset

You can load a time series dataset as a Pandas `Series` and specify the header row at line zero, as follows:

```
from pandas import read_csv
series = read_csv('daily-births.csv', header=0, index_col=0, parse_dates=True, squeeze=True)
```

Listing 3: Load time series dataset.

Get used to exploring loaded time series data in Python:

1. Print the first few rows using the `head()` function.
2. Print the dimensions of the dataset using the size attribute.
3. Query the dataset using a date-time string.
4. Print summary statistics of the observations.

For more help, see the post:

- How to Load and Explore Time Series Data in Python

In the next lesson, you will discover how to visualize time series data.

Lesson 03: Data Visualization

Data visualization is a big part of time series forecasting. Line plots of observations over time are popular, but there is a suite of other plots that you can use to learn more about your problem. In this lesson, you must download a standard time series dataset and create 6 different types of plots.

Download the monthly shampoo sales dataset from DataMarket in CSV format and save it with the filename `shampoo-sales.csv`.

- Download Dataset

Now create the following 6 types of plots:

1. Line Plots.
2. Histograms and Density Plots.
3. Box and Whisker Plots by year or quarter.
4. Heat Maps.
5. Lag Plots or Scatter Plots.
6. Autocorrelation Plots.

Below is an example of a simple line plot to get you started:

```
from pandas import read_csv
from matplotlib import pyplot
series = read_csv('shampoo-sales.csv', header=0, index_col=0, parse_dates=True,
    squeeze=True)
series.plot()
pyplot.show()
```

Listing 4: Load dataset and graph with line plot.

For more help, see the post:

- Time Series Data Visualization with Python

In the next lesson, you will discover how to develop a persistence forecast.

Lesson 04: Persistence Forecast Model

It is important to establish a baseline forecast. The simplest forecast you can make is to use the current observation (`t`) to predict the observation at the next time step (`t+1`). This is called the naive forecast or the persistence forecast and may be the best possible model on some time series forecast problems. In this lesson, you will make a persistence forecast for a standard time series forecast problem.

Download the daily female births dataset from DataMarket in CSV format and save it with the filename `daily-births.csv`.

- Download Dataset

You can implement the persistence forecast as a single line function, as follows:

```
# persistence model
def model_persistence(x):
  return x
```

Listing 5: Persistence forecast model.

Write code to load the dataset and use the persistence forecast to make a prediction for each time step in the dataset. Note, that you will not be able to make a forecast for the first time step in the dataset as there is no previous observation to use. Store all of the predictions in a list. You can calculate a Root Mean Squared Error (RMSE) for the predictions compared to the actual observations as follows:

```
from sklearn.metrics import mean_squared_error
from math import sqrt
predictions = []
actual = series.values[1:]
rmse = sqrt(mean_squared_error(actual, predictions))
```

Listing 6: Calculate RMSE for time series forecast.

For more help, see the post:

- How to Make Baseline Predictions for Time Series Forecasting with Python

In the next lesson, you will discover how to develop an autoregressive forecast model.

Lesson 05: Autoregressive Forecast Model

Autoregression means developing a linear model that uses observations at previous time steps to predict observations at future time step (*auto* means self in ancient Greek). Autoregression is a quick and powerful time series forecasting method. The Statsmodels Python library provides the autoregression model in the AR class[1]. In this lesson, you will develop an autoregressive forecast model for a standard time series dataset.

Download the monthly shampoo sales dataset from DataMarket in CSV format and save it with the filename `shampoo-sales.csv`.

- Download Dataset

You can fit an AR model as follows:

```
model = AR(dataset)
model_fit = model.fit()
```

Listing 7: Example of fitting an AR model.

You can predict the next out of sample observation with a fit AR model as follows:

```
prediction = model_fit.predict(start=len(dataset), end=len(dataset))
```

Listing 8: Example of making a prediction with an AR model.

You may want to experiment by fitting the model on half of the dataset and predicting one or more of the second half of the series, then compare the predictions to the actual observations. For more help, see the post:

- Autoregression Models for Time Series Forecasting With Python

In the next lesson, you will discover the ARIMA forecast model.

[1] http://statsmodels.sourceforge.net/stable/generated/statsmodels.tsa.ar_model.AR.html

Lesson 06: ARIMA Forecast Model

The ARIMA is a classical linear model for time series forecasting. It combines the autoregressive model (AR), differencing to remove trends and seasonality, called integrated (I) and the moving average model (MA) which is an old name given to a model that forecasts the error, used to correct predictions. The Statsmodels Python library provides the ARIMA class[2]. In this lesson, you will develop an ARIMA model for a standard time series dataset.

Download the monthly shampoo sales dataset from DataMarket in CSV format and save it with the filename `shampoo-sales.csv`.

- Download Dataset

The ARIMA class requires an `order(p,d,q)` that is comprised of three arguments `p`, `d` and `q` for the AR lags, number of differences and MA lags. You can fit an ARIMA model as follows:

```
model = ARIMA(dataset, order=(0,1,0))
model_fit = model.fit()
```

Listing 9: Example of fitting an ARIMA model.

You can make a one-step out-of-sample forecast for a fit ARIMA model as follows:

```
outcome = model_fit.forecast()[0]
```

Listing 10: Example of making a prediction with an ARIMA model.

The shampoo dataset has a trend so I'd recommend a `d` value of 1. Experiment with different `p` and `q` values and evaluate the predictions from resulting models. For more help, see the post:

- How to Create an ARIMA Model for Time Series Forecasting with Python

In the next lesson, you will tie all of the lessons together into an end-to-end project.

[2] http://statsmodels.sourceforge.net/stable/generated/statsmodels.tsa.arima_model.ARIMA.html

Lesson 07: Hello World End-to-End Project

You now have the tools to work through a time series problem and develop a simple forecast model. In this lesson, you will use the skills learned from all of the prior lessons to work through a new time series forecasting problem.

Download the quarterly S&P 500 index, 1900-1996 dataset from DataMarket in CSV format and save it with the filename `sp500.csv`.

- Download Dataset

Split the data, perhaps extract the last 4 or 8 quarters to a separate file. Work through the problem and develop forecasts for the missing data, including:

1. Load and explore the dataset.
2. Visualize the dataset.
3. Develop a persistence model.
4. Develop an autoregressive model.
5. Develop an ARIMA model.
6. Visualize forecasts and summarize forecast error.

For an example of working through a project, see the post:

- Time Series Forecast Study with Python: Monthly Sales of French Champagne

Final Word Before You Go...

You made it. Well done! Take a moment and look back at how far you have come. You discovered:

- How to frame a time series forecasting problem as supervised learning.
- How to load and explore time series data with Pandas.
- How to plot and visualize time series data a number of different ways.
- How to develop a naive forecast called the persistence model as a baseline.
- How to develop an autoregressive forecast model using lagged observations.
- How to develop an ARIMA model including autoregression, integration and moving average elements.
- How to pull all of these elements together into an end-to-end project.

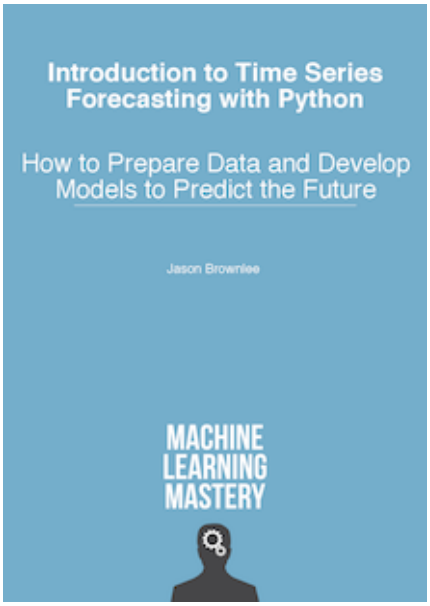
Don't make light of this, you have come a long way in a short amount of time. This is just the beginning of your time series forecasting journey with Python. Keep practicing and developing your skills.

How Did You Go With The Mini-Course?

Did you enjoy this mini-course?
Do you have any questions or sticking points?

Let me know, send me an email at: **jason@MachineLearningMastery.com**

If you would like me to step you through each lesson in great detail (and much more), take a look at my book: **Introduction to Time Series Forecasting with Python:**



Learn more here:

- Introduction to Time Series Forecasting with Python