Autoregression Models for Time Series Forecasting With Python

by **Jason Brownlee** on January 2, 2017 in **Time Series**

Autoregression is a time series model that uses observations from previous time steps as input to a regression equation to predict the value at the next time step.

It is a very simple idea that can result in accurate forecasts on a range of time series problems.

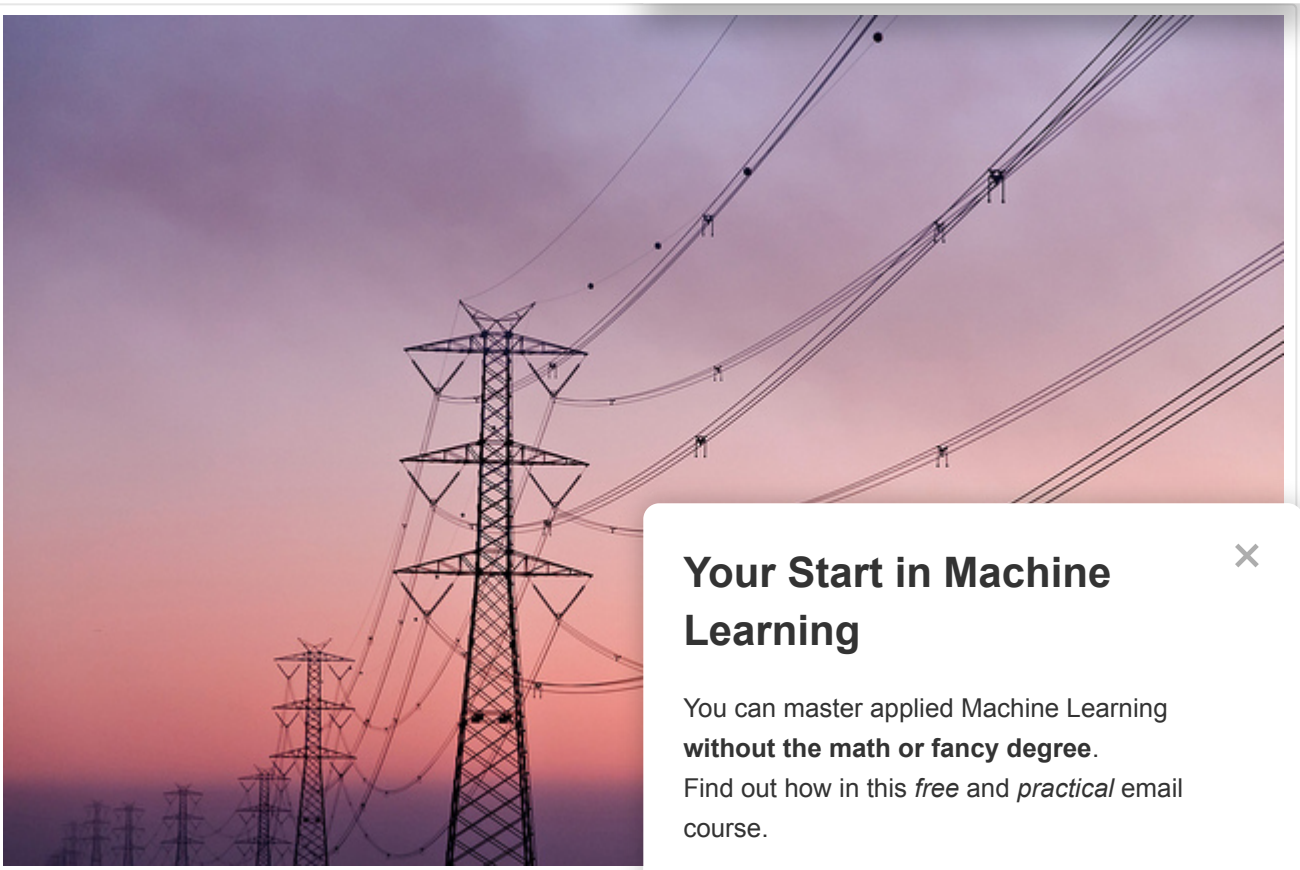
In this tutorial, you will discover how to implement an autoregressive model for time series forecasting with Python.

After completing this tutorial, you will know:

- How to explore your time series data for autocorrelation.
- How to develop an autocorrelation model and use it to make predictions.
- How to use a developed autocorrelation model to make rolling predictions.

Let's get started.

- **Update May/2017**: Fixed small typo in autoregression equation.



Autoregression Models for Time
Photo by Umberto Salvag

Autoregression

A regression model, such as linear regression, models an output value based on a linear combination of input values.

For example:

```
1 yhat = b0 + b1*X1
```

Where yhat is the prediction, b0 and b1 are coefficients found by optimizing the model on training data, and X is an input value.

This technique can be used on time series where input variables are taken as observations at previous time steps, called lag variables.

For example, we can predict the value for the next time step (t+1) given the observations at the last two time steps (t-1 and t-2). As a regression model, this would look as follows:

```
1 X(t+1) = b0 + b1*X(t-1) + b2*X(t-2)
```

Because the regression model uses data from the same input variable at previous time steps, it is referred to as an autoregression (regression of self).

Autocorrelation

An autoregression model makes an assumption that t to predict the value at the next time step.

This relationship between variables is called correlatic

If both variables change in the same direction (e.g. go positive correlation. If the variables move in opposite d one goes down), then this is called negative correlatio

We can use statistical measures to calculate the corre previous time steps at various different lags. The stron nd a specific lagged variable, the more weight that autoreg modeling.

Again, because the correlation is calculated between the variable and itself at previous time steps, it is called an autocorrelation. It is also called serial correlation because of the sequenced structure of time series data.

The correlation statistics can also help to choose which lag variables will be useful in a model and which will not.

Interestingly, if all lag variables show low or no correlation with the output variable, then it suggests that the time series problem may not be predictable. This can be very useful when getting started on a new dataset.

In this tutorial, we will investigate the autocorrelation of a univariate time series then develop an autoregression model and use it to make predictions.

Before we do that, let's first review the Minimum Daily Temperatures data that will be used in the examples.

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.

Learn more about the dataset here.

Download the dataset into 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 code below will load the dataset as a Pandas Ser

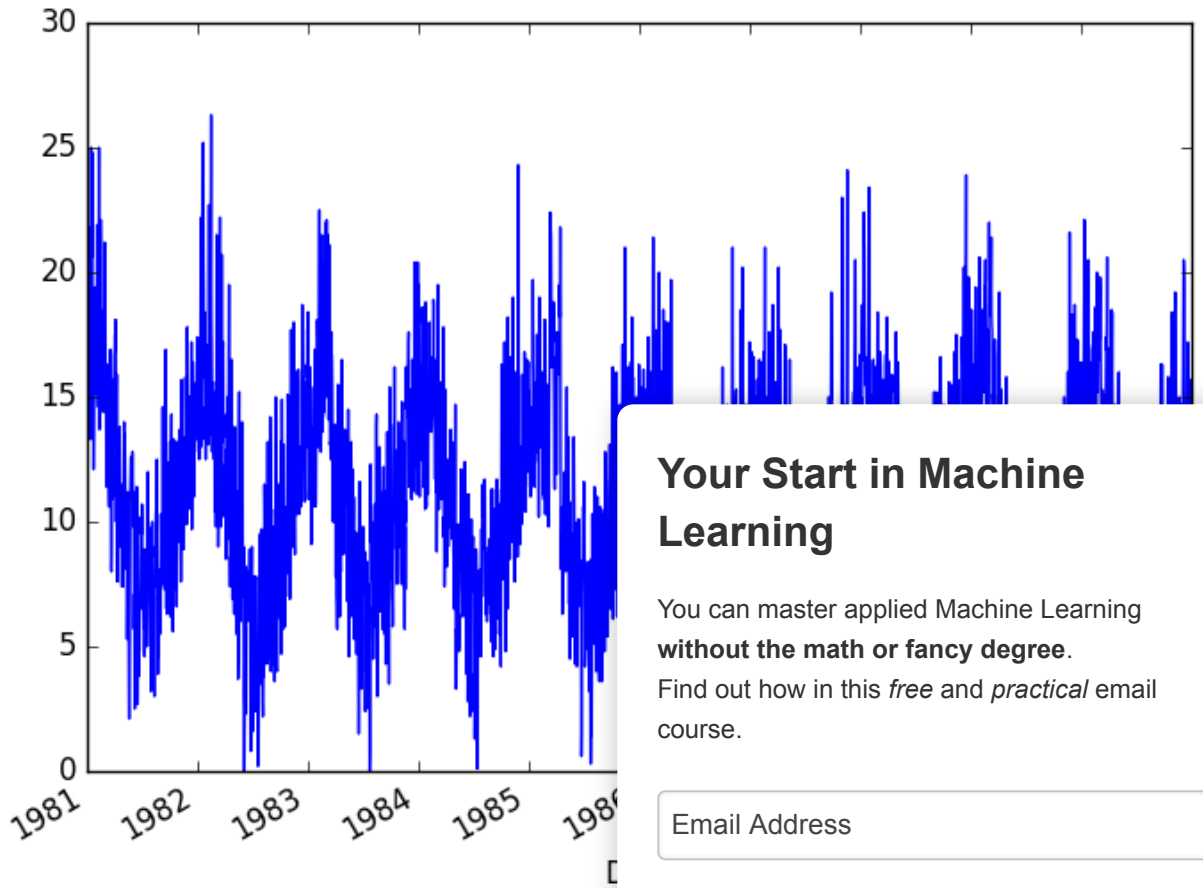
```
from pandas import Series
from matplotlib import pyplot
series = Series.from_csv('daily-minimum-temper
print(series.head())
series.plot()
pyplot.show()
```

Running the example prints the first 5 rows from the lo

```
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
```

A line plot of the dataset is then created.

Your Start in Machine Learning



Minimum Daily Temperature Dataset Plot

Quick Check for Autocorrelation

There is a quick, visual check that we can do to see if there is an autocorrelation in our time series dataset.

We can plot the observation at the previous time step (t-1) with the observation at the next time step (t+1) as a scatter plot.

This could be done manually by first creating a lag version of the time series dataset and using a built-in scatter plot function in the Pandas library.

But there is an easier way.

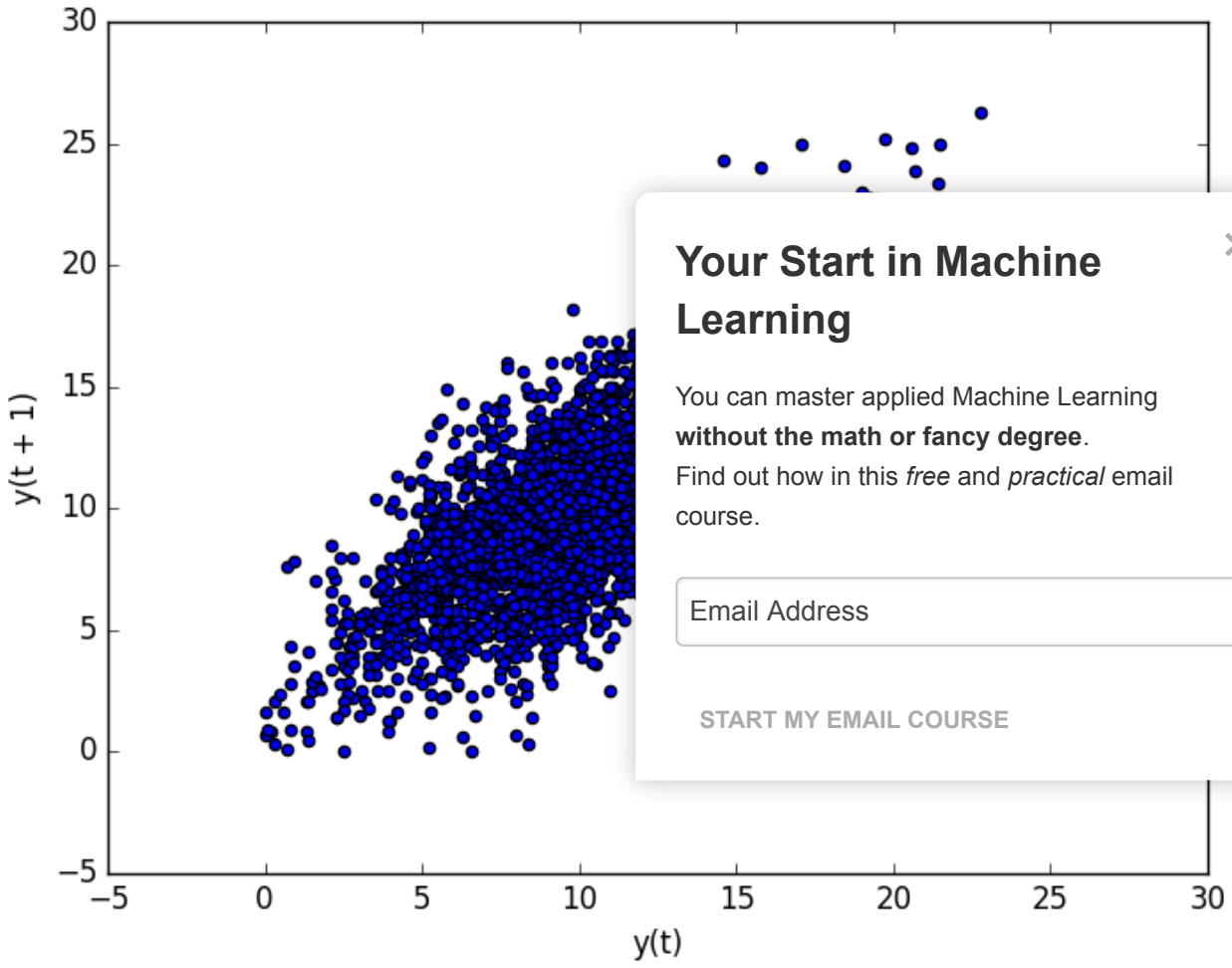
Pandas provides a built-in plot to do exactly this, called the lag_plot() function.

Below is an example of creating a lag plot of the Minimum Daily Temperatures dataset.

```
1 from pandas import Series
2 from matplotlib import pyplot
3 from pandas.tools.plotting import lag_plot
4 series = Series.from_csv('daily-minimum-temper
```

```
5 lag_plot(series)
6 pyplot.show()
```

Running the example plots the temperature data (t) on the x-axis against the temperature on the previous day (t-1) on the y-axis.



Minimum Daily Temperature Dataset Lag Plot

We can see a large ball of observations along a diagonal line of the plot. It clearly shows a relationship or some correlation.

This process could be repeated for any other lagged observation, such as if we wanted to review the relationship with the last 7 days or with the same day last month or last year.

Another quick check that we can do is to directly calculate the correlation between the observation and the lag variable.

We can use a statistical test like the Pearson correlation coefficient. This produces a number to summarize how correlated two variables are between -1 (negatively correlated) and +1 (positively correlated) with small values close to zero indicating low correlation and high values above 0.5 or below -0.5 showing high correlation.

Correlation can be calculated easily using the corr() function on the DataFrame of the lagged dataset.

The example below creates a lagged version of the Minimum Daily Temperatures dataset and calculates a correlation matrix of each column with other columns, including itself.

```
from pandas import Series
from pandas import DataFrame
from pandas import concat
from matplotlib import pyplot
series = Series.from_csv('daily-minimum-temperatures.csv', header=0)
values = DataFrame(series.values)
dataframe = concat([values.shift(1), values], axis=1)
dataframe.columns = ['t-1', 't+1']
result = dataframe.corr()
print(result)
```

This is a good confirmation for the plot above.

It shows a strong positive correlation (0.77) between t

```
          t-1      t+1
t-1   1.00000  0.77487
t+1   0.77487  1.00000
```

This is good for one-off checks, but tedious if we want ne series.

Next, we will look at a scaled-up version of this approa

Autocorrelation Plots

We can plot the correlation coefficient for each lag variable.

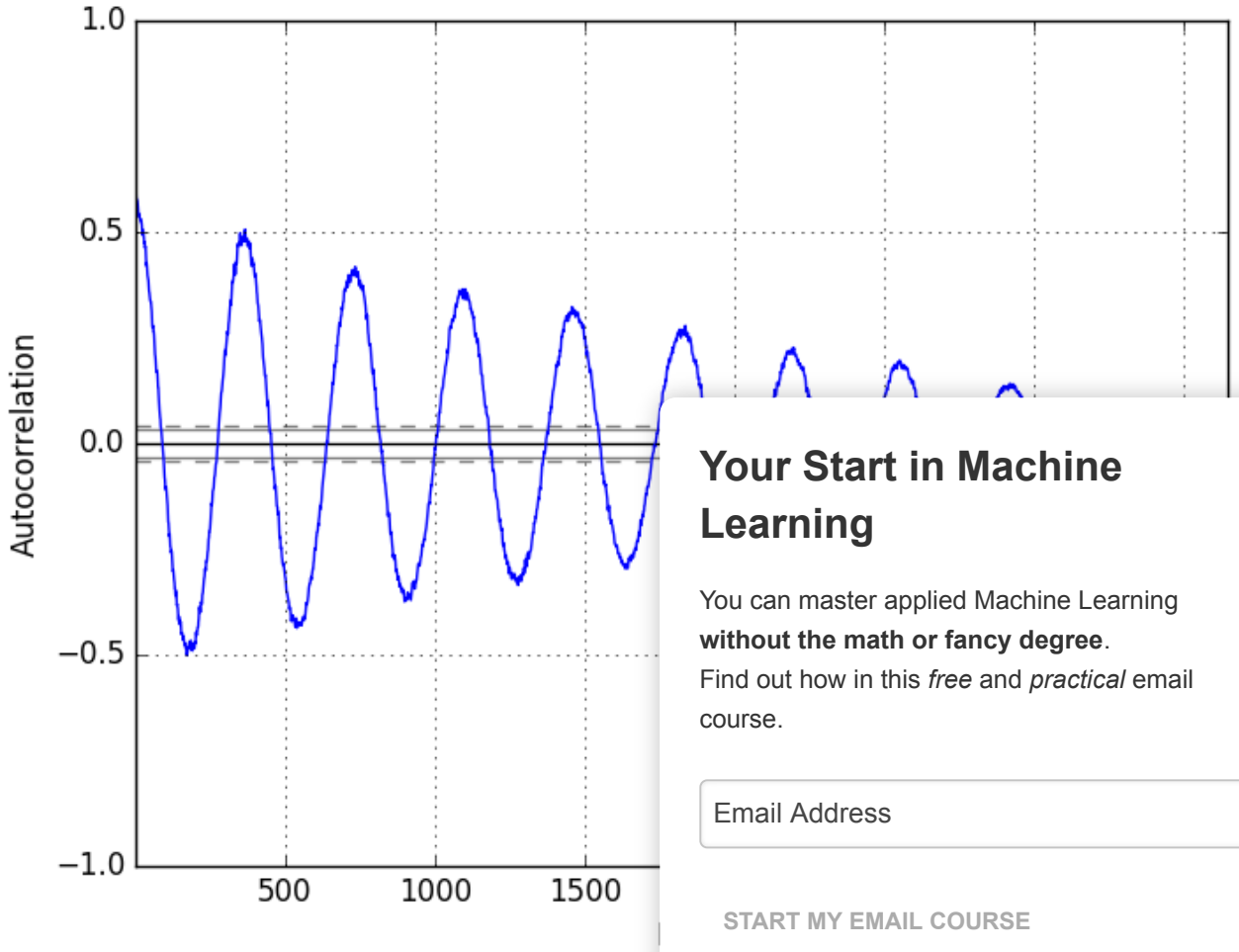
This can very quickly give an idea of which lag variables may be good candidates for use in a predictive model and how the relationship between the observation and its historic values changes over time.

We could manually calculate the correlation values for each lag variable and plot the result. Thankfully, Pandas provides a built-in plot called the autocorrelation_plot() function.

The plot provides the lag number along the x-axis and the correlation coefficient value between -1 and 1 on the y-axis. The plot also includes solid and dashed lines that indicate the 95% and 99% confidence interval for the correlation values. Correlation values above these lines are more significant than those below the line, providing a threshold or cutoff for selecting more relevant lag values.

```
from pandas import Series
from matplotlib import pyplot
from pandas.tools.plotting import autocorrelation_plot
series = Series.from_csv('daily-minimum-temperatures.csv', header=0)
autocorrelation_plot(series)
pyplot.show()
```

Running the example shows the swing in positive and negative correlation as the temperature values change across summer and winter seasons each prev

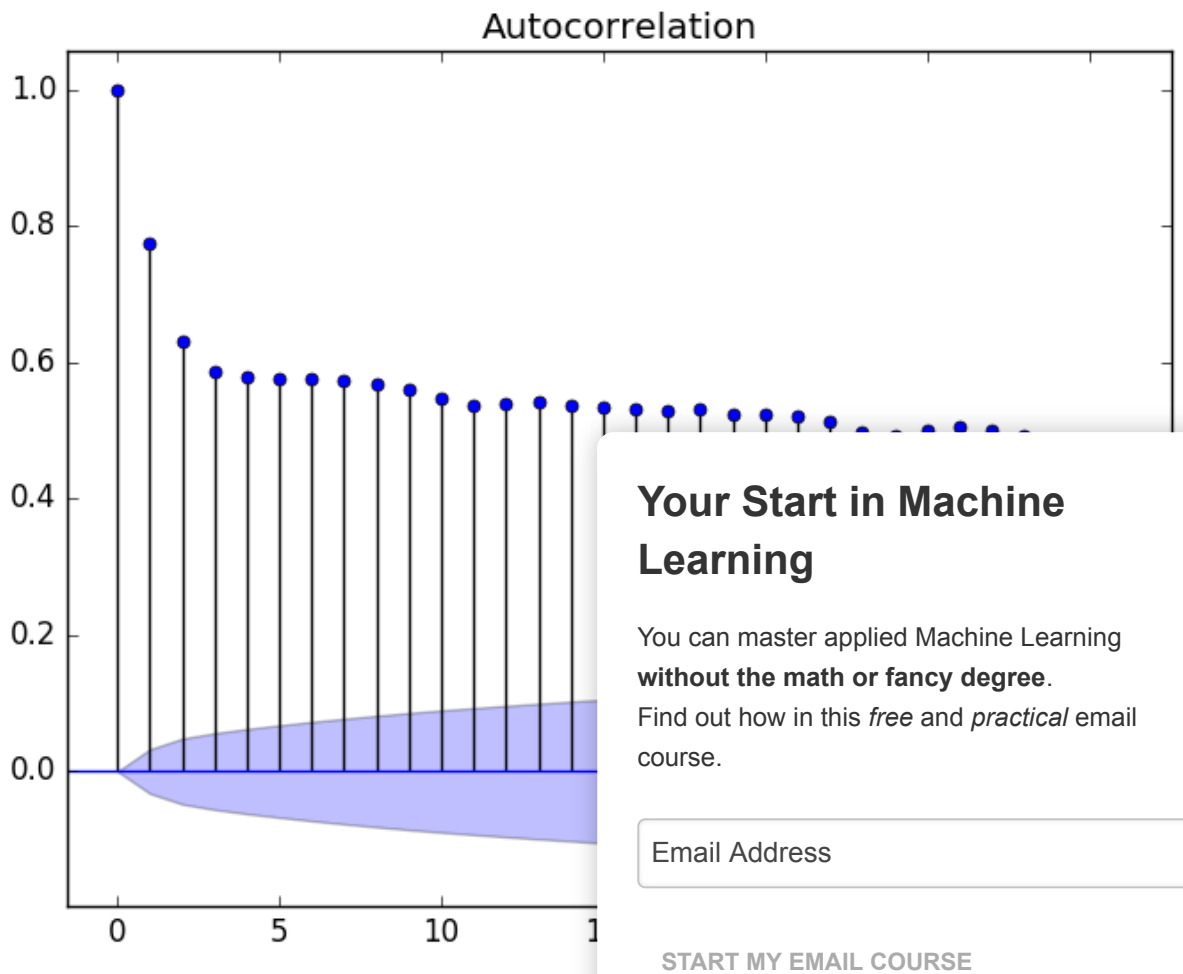


Pandas Autocorrelation Plot

The statsmodels library also provides a version of the plot in the plot_acf() function as a line plot.

```
1 from pandas import Series
2 from matplotlib import pyplot
3 from statsmodels.graphics.tsaplots import plot_acf
4 series = Series.from_csv('daily-minimum-temperatures.csv', header=0)
5 plot_acf(series, lags=31)
6 pyplot.show()
```

In this example, we limit the lag variables evaluated to 31 for readability.



Statsmodels Autocorrelation Plot

Now that we know how to review the autocorrelation in our time series, let's look at modeling it with an autoregression.

Before we do that, let's establish a baseline performance.

Persistence Model

Let's say that we want to develop a model to predict the last 7 days of minimum temperatures in the dataset given all prior observations.

The simplest model that we could use to make predictions would be to persist the last observation. We can call this a persistence model and it provides a baseline of performance for the problem that we can use for comparison with an autoregression model.

We can develop a test harness for the problem by splitting the observations into training and test sets, with only the last 7 observations in the dataset assigned to the test set as "unseen" data that we wish to predict.

The predictions are made using a walk-forward validation model so that we can persist the most recent observations for the next day. This means that we are not making a 7-day forecast, but 7 1-day forecasts.

```
from pandas import Series
from pandas import DataFrame
from pandas import concat
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
series = Series.from_csv('daily-minimum-temperatures.csv', header=0)
# create lagged dataset
values = DataFrame(series.values)
dataframe = concat([values.shift(1), values], axis=1)
dataframe.columns = ['t-1', 't+1']
# split into train and test sets
X = dataframe.values
train, test = X[1:len(X)-7], X[len(X)-7:]
train_X, train_y = train[:,0], train[:,1]
test_X, test_y = test[:,0], test[:,1]

# persistence model
def model_persistence(x):
    return x

# walk-forward validation
predictions = list()
for x in test_X:
    yhat = model_persistence(x)
    predictions.append(yhat)
test_score = mean_squared_error(test_y, predi
print('Test MSE: %.3f' % test_score)
# plot predictions vs expected
pyplot.plot(test_y)
pyplot.plot(predictions, color='red')
pyplot.show()
```

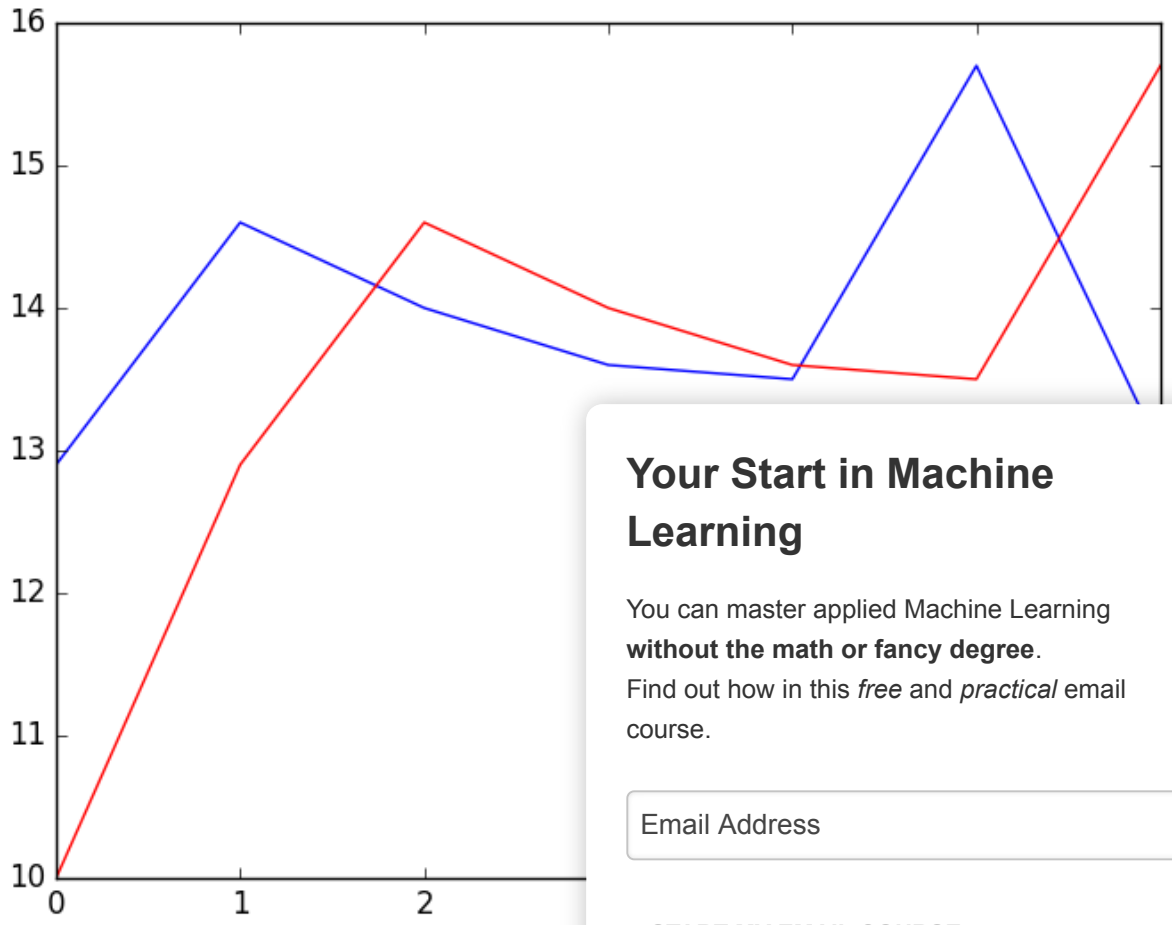

Running the example prints the mean squared error (MSE).

The value provides a baseline performance for the problem.

```
Test MSE: 3.423
```

The expected values for the next 7 days are plotted (blue) compared to the predictions from the model (red).

Your Start in Machine Learning



Predictions From Persistence Model

Autoregression Model

An autoregression model is a linear regression model that uses lagged variables as input variables.

We could calculate the linear regression model manually using the LinearRegession class in scikit-learn and manually specify the lag input variables to use.

Alternately, the statsmodels library provides an autoregression model that automatically selects an appropriate lag value using statistical tests and trains a linear regression model. It is provided in the AR class.

We can use this model by first creating the model AR() and then calling fit() to train it on our dataset. This returns an ARResult object.

Once fit, we can use the model to make a prediction by calling the predict() function for a number of observations in the future. This creates 1 7-day forecast, which is different from the persistence example above.

The complete example is listed below.

```
from pandas import Series
from matplotlib import pyplot
from statsmodels.tsa.ar_model import AR
from sklearn.metrics import mean_squared_error
series = Series.from_csv('daily-minimum-temperatures.csv', header=0)
# split dataset
X = series.values
train, test = X[1:len(X)-7], X[len(X)-7:]
# train autoregression
model = AR(train)
model_fit = model.fit()
print('Lag: %s' % model_fit.k_ar)
print('Coefficients: %s' % model_fit.params)
# make predictions
predictions = model_fit.predict(start=len(tra
for i in range(len(predictions)):
    print('predicted=%f, expected=%f' % (pred
error = mean_squared_error(test, predictions)
print('Test MSE: %.3f' % error)
# plot results
pyplot.plot(test)
pyplot.plot(predictions, color='red')
pyplot.show()
```


Running the example first prints the chosen optimal la
regression model.

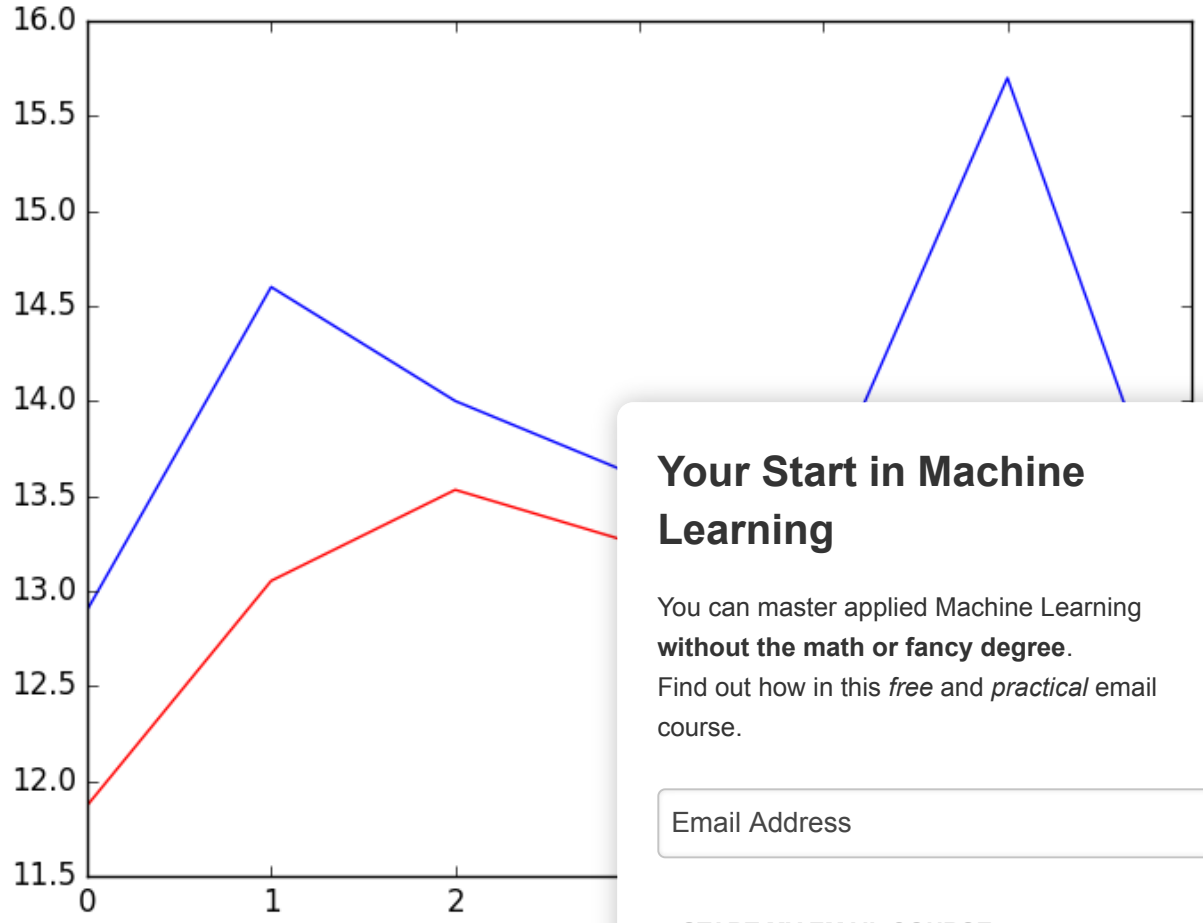
We can see that a 29-lag model was chosen and train
the average number of days in a month.

The 7 day forecast is then printed and the mean squa

```
Lag: 29
Coefficients: [  5.57543506e-01   5.88595221e-01  -9.08257090e-02   4.82615092e-02
   4.00650265e-02   3.93020055e-02   2.59463738e-02   4.46675960e-02
   1.27681498e-02   3.74362239e-02  -8.11700276e-04   4.79081949e-03
   1.84731397e-02   2.68908418e-02   5.75906178e-04   2.48096415e-02
   7.40316579e-03   9.91622149e-03   3.41599123e-02  -9.11961877e-03
   2.42127561e-02   1.87870751e-02   1.21841870e-02  -1.85534575e-02
  -1.77162867e-03   1.67319894e-02   1.97615668e-02   9.83245087e-03
   6.22710723e-03  -1.37732255e-03]
predicted=11.871275, expected=12.900000
predicted=13.053794, expected=14.600000
predicted=13.532591, expected=14.000000
predicted=13.243126, expected=13.600000
predicted=13.091438, expected=13.500000
predicted=13.146989, expected=15.700000
predicted=13.176153, expected=13.000000
Test MSE: 1.502
```

A plot of the expected (blue) vs the predicted values (red) is made.

The forecast does look pretty good (about 1 degree Celsius out each day), with big deviation on day 5.



Predictions From Fixed AR Model

The statsmodels API does not make it easy to update the model as new observations become available.

One way would be to re-train the AR model each day as new observations become available, and that may be a valid approach, if not computationally expensive.

An alternative would be to use the learned coefficients and manually make predictions. This requires that the history of 29 prior observations be kept and that the coefficients be retrieved from the model and used in the regression equation to come up with new forecasts.

The coefficients are provided in an array with the intercept term followed by the coefficients for each lag variable starting at t-1 to t-n. We simply need to use them in the right order on the history of observations, as follows:

```
yhat = b0 + b1*X1 + b2*X2 ... bn*Xn
```

Below is the complete example.

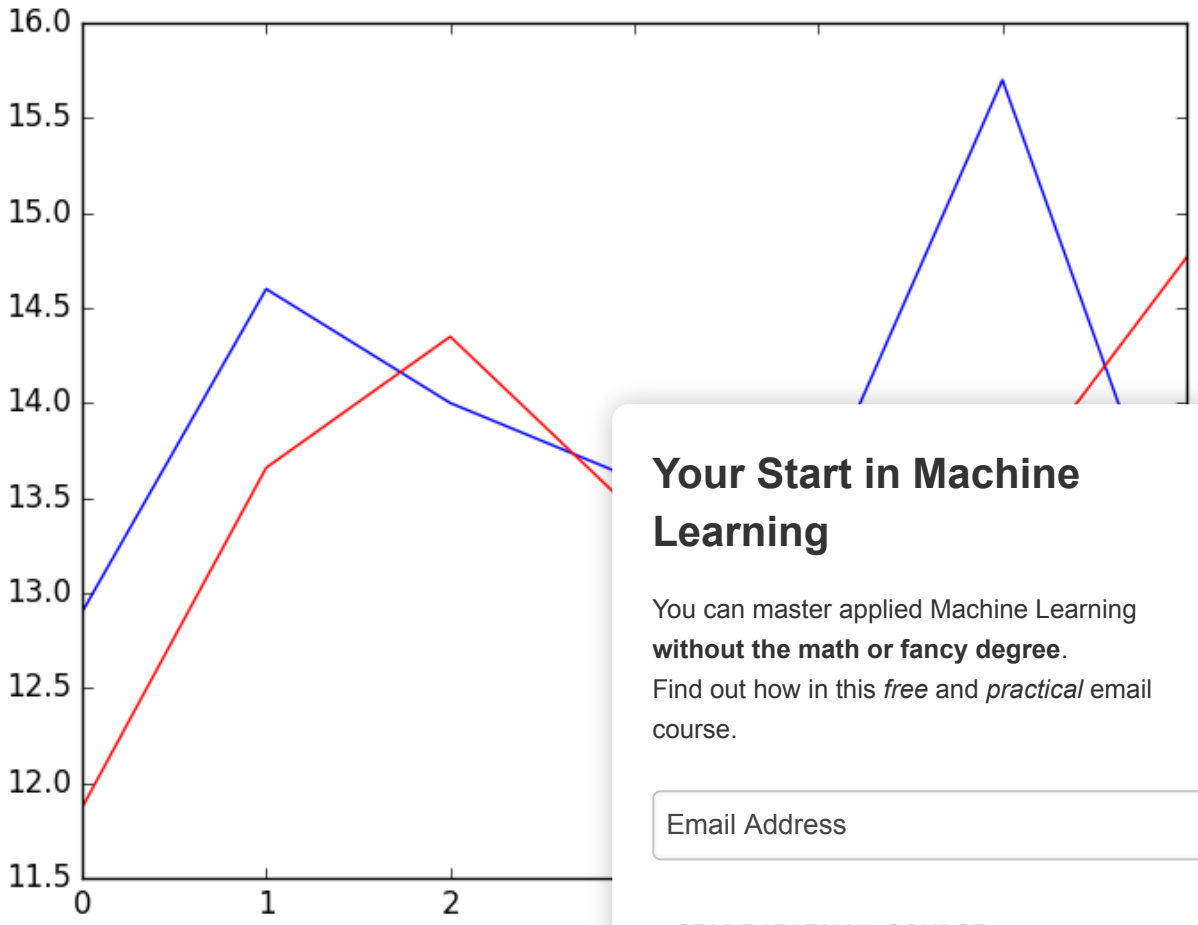
```
from pandas import Series
from matplotlib import pyplot
from statsmodels.tsa.ar_model import AR
```

```
from sklearn.metrics import mean_squared_error
series = Series.from_csv('daily-minimum-temperatures.csv', header=0)
# split dataset
X = series.values
train, test = X[1:len(X)-7], X[len(X)-7:]
# train autoregression
model = AR(train)
model_fit = model.fit()
window = model_fit.k_ar
coef = model_fit.params
# walk forward over time steps in test
history = train[len(train)-window:]
history = [history[i] for i in range(len(history))]
predictions = list()
for t in range(len(test)):
    length = len(history)
    lag = [history[i] for i in range(length-window,length)]
    yhat = coef[0]
    for d in range(window):
        yhat += coef[d+1] * lag[window-d-1]
    obs = test[t]
    predictions.append(yhat)
    history.append(obs)
    print('predicted=%f, expected=%f' % (yhat
error = mean_squared_error(test, predictions)
print('Test MSE: %.3f' % error)
# plot
pyplot.plot(test)
pyplot.plot(predictions, color='red')
pyplot.show()
```


Again, running the example prints the forecast and the

```
predicted=11.871275, expected=12.900000
predicted=13.659297, expected=14.600000
predicted=14.349246, expected=14.000000
predicted=13.427454, expected=13.600000
predicted=13.374877, expected=13.500000
predicted=13.479991, expected=15.700000
predicted=14.765146, expected=13.000000
Test MSE: 1.451
```

We can see a small improvement in the forecast when comparing the error scores.



Predictions From Rolling AR Model

Further Reading

This section provides some resources if you are looking to dig deeper into autocorrelation and autoregression.

- Autocorrelation on Wikipedia
- Autoregressive model on Wikipedia
- Chapter 7 – Regression-Based Models: Autocorrelation and External Information, Practical Time Series Forecasting with R: A Hands-On Guide.
- Section 4.5 – Autoregressive Models, Introductory Time Series with R.

Summary

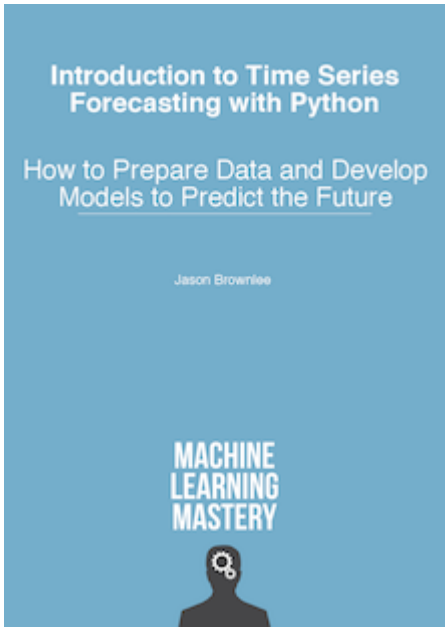
In this tutorial, you discovered how to make autoregression forecasts for time series data using Python.

Specifically, you learned:

- About autocorrelation and autoregression and how they can be used to better understand time series data.
- How to explore the autocorrelation in a time series using plots and statistical tests.
- How to train an autoregression model in Python and use it to make short-term and rolling forecasts.

Do you have any questions about autoregression, or about this tutorial?
Ask your questions in the comments below and I will do my best to answer.

Want to Develop Time Series Forecasts with Python?



Deve

Introdu

It covers **self-s**
Loading data, vis

Finall

About Jason Brownlee

Jason Brownlee, Ph.D. is a machine learning specialist who teaches developers how to get results with modern machine learning methods via hands-on tutorials.
View all posts by Jason Brownlee →

‹ How to Check if Time Series Data is Stationary with Python

Time Series Data Visualization with Python ›

82 Responses to *Autoregression Models for Time Series Forecasting With Python*

Gary Bake January 5, 2017 at 11:35 pm #

REPLY

Thank you Jason for the awesome article

In case anyone hits the same problem I had –
I downloaded the data from the link above as a csv file.
It was failing to be imported due to three rows in the temperature column containing '?'.
Once these were removed the data imported ok.

Jason Brownlee January 6, 2017 at 9:10 am #

REPLY

Thanks for the heads up Gary.

Tim Melino January 14, 2017 at 10:28 am #

Hey Jason, thanks for the article. How would
expected value is not known?

Jason Brownlee January 15, 2017 at 5:27 am

Hi Tim, you can use mode.predict() as in
be predicted.

Siddhant Bhambri October 16, 2018 at 12:47 am #

REPLY

Hey Jason, I am not clear with the use of model.predict(). If you could help me on this in predicting the values for the next 10 days if the model has learned the values till today.

Jason Brownlee October 16, 2018 at 6:39 am #

REPLY

Sure, see this post:
https://machinelearningmastery.com/make-sample-forecasts-arima-python/

Farrukh Jalali January 25, 2017 at 4:16 pm #

REPLY

Hi Jason,

Thanks for all of your wonderful blogs. They are really helping a lot. One question regarding this post is that I believe that AR modeling also presume that time series is stationary as the observations should be i.i.d. .Does that AR function from statsmodels library checks for stationary and use the de-trended de-seasonalized time series by itself if required? Also, if we use sckit learn library for AR model as you described do we need to check for and make adjustments by ourselfs for this?

Jason Brownlee January 26, 2017 at 4:45 am #

REPLY

Hi Farrukh, great question.

The AR in statsmodels does assume that the data is stationary.

If your data is not stationary, you must make it stat

Farrukh Jalali January 31, 2017 at 12:19

Thanks for the answer. Though we di
stationarity check in the example above but fr
seasonal effect. So in that case whether apply

Jason Brownlee February 1, 2017

Great question Farrukh.

AR is designed to be used on stationary data, meaning data with no seasonal or trend information.

Farrukh Jalali January 31, 2017 at 11:07 pm #

REPLY

Or to be specific, is it OK to apply AR model direct here on the given data without checking the seasonality and removing it if present which is showing some signs in first graph apparently?

Anthony of Sydney March 8, 2017 at 5:09 am #

REPLY

Dear Dr Jason,
I had a go at the ‘persistence model’ section.
The dataset I used was the sp500.csv dataset.
From your code

```
1  # persistence model
```

Your Start in Machine Learning

```
def model_persistence(x):
    return x

# walk-forward validation
predictions = list()
for x in test_X:
    yhat = model_persistence(x)
    predictions.append(yhat)
test_score = mean_squared_error(test_y, predictions)
```

As soon as I try to compute the “test_score”, I get the following error,

```
Traceback (most recent call last):
  File "", line 1, in
    test_score = mean_squared_error(test_y,predictions)
  File "C:\Python34\lib\site-packages\sklearn\metrics\regression.py", line 232, in mean_sq
    output_errors = np.average((y_true - y_pred) ** 2, axis=0
TypeError: ufunc 'subtract' did not contain
```

Any idea,
Anthony of Sydney NSW

Jason Brownlee March 8, 2017 at 9:46 am #

It looks like you need to convert your data

E.g. in numpy this might be:

```
X = X.astype('float32')
```

Anthony of Sydney who noticed that it was a string not a float March 8, 2017 at 5:30 REPLY

Dear Dr Jason,
Fixed the problem. You cannot assume that all *.csv numbers are floats or ints. For some reason, the numbers seem to be enclosed in quotes. Note that the data is is for the sp500.csv not the above exercise. Note the numbers in the output are enclosed in quotes:

```
test_y
array(['1.331', '1.412', '1.474', '1.559', '1.585', '1.788', '1.817'], dtype=object)
>>> predictions
['1.24', '1.331', '1.412', '1.474', '1.559', '1.585', '1.788']
>>> test_y = [float(i) for i in test_y] # convert string to float
>>> predictions = [float(i) for i in predictions]  # convert string to float
>>> test_score = mean_squared_error(test_y,predictions)
>>> test_score
0.009805285714285716
```

It works now,
Regards
Anthony from Sydney

Jason Brownlee March 8, 2017 at 9:46 am #

REPLY

Glad to hear you fixed it Anthony.

Anthony from Sydney March 8, 2017 at 6:07 am #

REPLY

Dear Dr Jason,
The problem has been fixed. The values in the array were strings, so I had to convert them to strings.

```
1 test_y
2 array(['1.331', '1.412', '1.474', '1.559',
3 >>> predictions
4 ['1.24', '1.331', '1.412', '1.474', '1.559'
```

So I converted the strings in each array to float.

```
1 >>> test_y = [float(i) for i in test_y]
2 >>> predictions = [float(i) for i in predict
3 >>> test_score = mean_squared_error(test_y,p
4 >>> test_score
5 0.009805285714285716
```

Hope that helps others trying to convert values to appr
calculations.
Anthony from Sydney NSW

Johnson S April 17, 2017 at 6:39 am #

REPLY

how can I show my prediction as date, instead of log, for example I have data set incident number for each week, I want to predict the following week
week1 669
week2 443
week3 555

so on april week 1 I want to show time series the prediction of week1 week2 of April

Jason Brownlee April 18, 2017 at 8:28 am #

REPLY

Sorry, I'm not sure I understand. Perhaps you can give a fuller example of inputs and outputs?

Chika April 30, 2017 at 2:46 am #

REPLY

Thanks for this wonderful tutorial. I discovered your articles on facebook last year. Since then I have been following all your tutorials and I must confess that, though I started learning about machine learning in less than a year, my knowledge base has tremendously increased as a result of this free services you have been posting for all to see on the website.
Thanks once more for this generosity Dr Jason.

My question is that, I have carefully followed all your procedures in this article on my data set that was recorded every 10mins for 2 months. Please I would like to know which time lag is appropriate for forecasting to see the next 7 days value or more or less.

My time series is a stationary one according to the test statistics and histogram I have applied on it. but I still don't know if a reasonable conclusion can be reached with a data set that was recorded for 2 months every 10mins daily.

Jason Brownlee April 30, 2017 at 5:33 am #

Well done on your progress!

This post gives you some ideas on how to select s
http://machinelearningmastery.com/gentle-introduc

I hope that helps as a start.

Soy May 14, 2017 at 4:35 am #

Dr Jason,
Thank you so much for this post. I have finally learned how to go from theory to practice.

Jason Brownlee May 14, 2017 at 7:33 am #

REPLY

I'm so glad to hear that Soy, thanks!

Akshit Mantri June 8, 2017 at 10:35 pm #

REPLY

Dr Jason, How do i predict the low and high confidence interval of prediction of an AR model?

Jason Brownlee June 9, 2017 at 6:25 am #

REPLY

This post will help:
http://machinelearningmastery.com/time-series-for

Akshit Mantri June 9, 2017 at 5:40 pm #

REPLY

Thanks for the reply. It was a very good post. But, it was for ARIMA model. I am having some problem with the ARIMA model I cannot use it. Is there such a confidence interval forecasting for AR model?

Jason Brownlee June 10, 2017 at 8:18 am #

REPLY

Yes, I believe the same approach

Daoud June 11, 2017 at 9:10 pm #

ValueError: On entry to DLASCL parameter n
I ma found above error when i use
model = AR(train) ## no error
model_fit = model.fit() ## show above error

Samuel N June 15, 2017 at 6:10 am #

Thanks for the great write-up Jason. One question though , I am interested in doing an autoregression on my timeseries data every nth days. For example. Picking the first 20 days and predicting the value of the 20th day. Then picking the next 20 days (shift 1 over) and predict the value of the 21st day and so forth until the end of my dataset. How can this be achieved in code? Thanks.

Jason Brownlee June 15, 2017 at 8:54 am #

REPLY

Consider using a variation of walk forward validation:
http://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/

Dhineshkumar June 15, 2017 at 5:49 pm #

REPLY

Hi Jason,

Thanks for your article. I have few doubts.

1) We do analysis on the autocorrelation plots and auto-correlation function only after making the time series stationary right?

2) For the time series above, the correlation value is maximum for lag=1. So is it like the value at t-1 is given more weight while doing Autoregression?

3) When the AR model says, for lag-29 model the MSE is minimum. Does it mean the regression model constructed with values from t to t-29 gave minimum MSE?

Please clarify.

Thank you

Jason Brownlee June 16, 2017 at 7:52 am #

1. Ideally, yes, analysis after the data is st

2. Yes.

3. Yes.

Dhineshkumar June 16, 2017 at 6:46 pm

Thanks.

Jason Brownlee June 17, 2017 at 7:24 am #

REPLY

You're welcome.

TaeWoo Kim June 20, 2017 at 10:10 am #

REPLY

Hey Jason. Thanks for the awesome tutorial.

in this line…

print('Lag: %s' % model_fit.k_ar)
print('Coefficients: %s' % model_fit.params)

What is the "layman" explanation of what lag and coeffcients are?

Is it that "lag" is what the AR model determined that the significance ends (i.e. after this number, the autocorrelation isn't "strong enough) and that the coeff. are the p-value of null hypotehsis on "intensity" of the autocorrelation?

Jason Brownlee June 21, 2017 at 8:08 am #

REPLY

Lag observations are observations at prior time steps.

Lag coefficients are the weightings on those lag observations.

TaeWoo Kim June 22, 2017 at 5:03 am #

REPLY

Thank you Jason. That answer has still gotten me more confused. 🙂

The section where you manually calculate the predictions…you're specifying history.append(obs)
So does this mean that the test data up to t0 is

In another words, this model can only predict 1

If I am wrong, how do I tweak the code to pred

TaeWoo Kim June 22, 2017 at 5:49 am #

i think i answered my own after readir
step-time-series-forecasting/

Example you gave with statsmodels.tsa.ar_mc

Jason Brownlee June 22, 2017 at 6:13 am #

REPLY

Correct, but the forecast() function for ARIMA (and maybe AR) can make a multi-step forecast. It might just not be very good.

Yigit Yazicioglu September 30, 2017 at 12:29 am #

REPLY

Dear Jason

Thanks for this wonderful guideline. I have a problem with my dataset and I am digging in time series modeling.

I try to model a physical model such that

y(t+1) = y(t) + f(a(t), b(t), c(t)) with AR models.

Actually problem is predict metal temperature using basic heat transfer where y means metal Temperature and f is a function of some sensors data like coolant mass flow rate, coolant temperaute, gas flowrate and gas temperatue.

Which model do you advise to use?

REPLY

Jason Brownlee September 30, 2017 at 7:43 am #

Sounds like a great problem. I would recommend testing a suite of different methods to see what works best for your specific data, given your specific modeling requirements.

I recommend this process generally:
https://machinelearningmastery.com/start-here/#process

Monika October 13, 2017 at 3:48 am #

Hi Jason…
I read your few articles and found very helpful. I am ve
meaning of these prediction points such as predicted=
it mean???

how does it help to understanding the prediction?

please post about cross regression also if you have po

Jason Brownlee October 13, 2017 at 5:50 am

Sorry, I don't follow. Perhaps you can rephrase your question?

REPLY

Shantanu October 16, 2017 at 6:30 am #

Thanks for the wonderful article.
I have a question: How can we make a prediction based on multiple columns by AR?
For an example, there are columns of
Date, Pricing, ABC, PQR
ABC, PQR contribute in predicting prices. So I want to predict pricing based on these columns as well.
Thank you

REPLY

Jason Brownlee October 16, 2017 at 3:46 pm #

It may be possible, but I do not have a worked example, sorry.

Ryan Berger October 22, 2017 at 4:00 am #

REPLY

How could you make this a Deep Autoregressive Network with Keras?

Jason Brownlee October 22, 2017 at 5:33 am #

REPLY

A deep MLP will do the trick!

Forouq November 10, 2017 at 3:28 am #

REPLY

Thank you for the amazing tutorial
I have a question though. According to Pandas' Autoc
when lag=1. But the AR model selects lag=29 to build
I checked this code on my dataset, and the autoregres
that lag=14 chosen by AR model. Can you explain this
relationship, thus, the autoregression which maps a lin
on the lag variable giving the maximum Pearson correl

Jason Brownlee November 10, 2017 at 10:37

Perhaps the method was confused by noi

Duplec November 19, 2017 at 12:45 am #

REPLY

What if time is also included along with the date?
1981-01-01,20.7 is like 1981-01-01 03:00:00,20.7

Jason Brownlee November 19, 2017 at 11:08 am #

REPLY

Sure.

sam November 26, 2017 at 11:46 am #

REPLY

Hi great tutorial. Just wanted to ask how would I change the order or lag in the code? Also if you had any tutorials for understanding how to use the statsmodels library

Jason Brownlee November 27, 2017 at 5:46 am #

REPLY

My book is the best source of material on the topic:
https://machinelearningmastery.com/introduction-to-time-series-forecasting-with-python/

You can either difference your code directly or use the d parameter in the ARIMA model to control the differencing order. I have tutorials on both, perhaps start here:
https://machinelearningmastery.com/start-here/#timeseries

At December 31, 2017 at 4:25 am #

REPLY

Thanks a lot! I ran into a problem while develo
dropped. I had to mention the frequency parameter ev

Jason Brownlee December 31, 2017 at 5:21

Interesting. Did it fix your issues?

Carolyn March 14, 2018 at 1:52 am #

Hi Jason,

Excellent article! Any chance of a blog post on how to do vector autoregression with big data? Sometimes an LSTM is overkill, and even a vanilla RNN can be overkill, so something with just plain old autoregression would be great.

The VAR package in Python does this, but it runs into memory issues very quickly with large, sparse datasets.

If you know of any other tools for vector autoregression, any insight you have would be appreciated!

Jason Brownlee March 14, 2018 at 6:31 am #

REPLY

Thanks for the suggestion Carolyn.

Harshil April 23, 2018 at 7:04 pm #

REPLY

Dear Jason,

Thank you so much for your wonderful article. I have a doubt regarding Data driven forecasting. I need to forecast appliance energy which depends upon 26 variables.I have data of appliance energy along with 26 variables of 3 months. With the help of 26 variables How can I forecast appliance energy for future?

Jason Brownlee April 24, 2018 at 6:27 am #

REPLY

Good question.

You can transform the data into a supervised learning problem and try a suite of machine learning algorithms:
https://machinelearningmastery.com/convert-time-series-supervised-learning-problem-python/

I hope to provide more information on this topic in

Praveen May 27, 2018 at 10:21 pm #

Hey Jason!
Thank you for the article. I have a doubt regarding this
not present in the dataset?

Jason Brownlee May 28, 2018 at 5:59 am #

Call model.predict() and specify the dates or index.

This tutorial will show you how:
https://machinelearningmastery.com/make-sample-forecasts-arima-python/

Ashish June 21, 2018 at 5:43 pm #

REPLY

can you please provide me a detaile example over VAR model?

Jason Brownlee June 22, 2018 at 6:03 am #

REPLY

I hope to cover the method in the future, thanks for the suggestion.

Bhaskar Sharan July 10, 2018 at 6:29 pm #

REPLY

Your Start in Machine Learning

```
from pandas import Series
from matplotlib import pyplot
series = Series.from_csv('G:\Study_Material\daily-minimum-temperatures.csv')
print(series.head())
series.plot()
pyplot.show()
```

ha an error as : Empty 'DataFrame': no numeric data to plot

how to resolve it sir?

Jason Brownlee July 11, 2018 at 5:53 am #

Perhaps double check you have loaded th

Vijay July 30, 2018 at 10:53 am #

Its a wonderful post that I came across and th
examples. I am new to machine learning and I have a
timeseries. I have events that can recur every day, we
this could be business meetings. Different meetings co
to use ARIMA for predicting the underlying pattern. My
virtual meetings based on their past history. Lets assu
would be an appropriate algorithm for predicting resou
history. I tried ARIMA based on this tutorial but the results weren't convincing. I wasn't sure if its appropriate to model this problem as a time series problem and is ARIMA a good choice for such problems.

Jason Brownlee July 30, 2018 at 2:16 pm #

REPLY

Perhaps try it as a starting point.

Tim Boons August 7, 2018 at 6:53 pm #

REPLY

I'm trying to work out o AR model to forecast a series using a lag of 192.

The series has a datapoint every 15 min but the you receive the data, the day after it was measured. So you have to forecast the next day (D+1) with data of the previous day (D-1) hence a lag of 192 in datapoints that are 15 min apart.

Is there a way to contrain the AR() function to all datapoints before t-192 ?

Jason Brownlee August 8, 2018 at 6:16 am #

REPLY

Perhaps fit a regularized linear regression model directly on your chosen lags?

Tim Boons August 14, 2018 at 10:14 pm #

REPLY

Thanks for your reply!

Were can I find some more information about a regularized lineair regression model ?

Jason Brownlee August 15, 2018 a

Any good book on machine learni
https://amzn.to/2KSoQ0a

Kamal Pokharel August 27, 2018 at 4:54 pm #

Thanks a lot for this lovely article

Jason Brownlee August 28, 2018 at 5:57 am #

REPLY

Thanks, I'm happy it helped.

Allison August 28, 2018 at 1:00 pm #

REPLY

Hi Dr. Brownlee,

Fantastic article — I'm following along step by step and it's helping.

at the step:
model.fit()

I get this error:
TypeError: Cannot find a common data type.

Where could this come from?

Jason Brownlee August 29, 2018 at 7:59 am

That is odd. Are you using the code and data from the tutorial?

Did you replace the “?” chars in the data file?

Phil September 29, 2018 at 4:49 am #

REPLY

Shouldn’t your first equation be:

X(t) = b0 + b1*X(t-1) + b2*X(t-2)

Instead of:

X(t+1) = b0 + b1*X(t-1) + b2*X(t-2)

Jason Brownlee September 29, 2018 at 6:38

Then where is “t”?

ML Uros October 12, 2018 at 2:12 am #

Hi Jason. Great job with your blog and this art
I was wondering if you could help me with the following
your test set and the AR model has a lower MSE for th
make the test set larger, say a couple hundred points, and make AR predictions, I get a higher MSE with the AR model. I know a couple hundred points means like a year of data points and the AR model could be updated in between to obtain better results, but I was just wondering what is your view on this matter. Does it mean that the AR model is not suitable for predictions too far in the future?
Also, if I use the AR model for predicting about 180 points, AR’s MSE value rises quite significantly, to roughly 9. Interestingly, if the test set is enlarged even more to about 350 points, MSE value falls to about 7. Persistence model’s MSE has lower variability. What does this changing MSE say about the data and applying AR to it?

Jason Brownlee October 12, 2018 at 6:42 am #

REPLY

The further you predict into the future, the worse the performance.

Dieu Do October 13, 2018 at 1:54 pm #

REPLY

Hello Dr. Jason Brownlee,

Thanks a lot for your excellent article.
I have a question related to predicted results.
Why the predicted solution at i-th point is very close the expected solution at (i-1)-th point ?
In all most your article, I have seen this.
I don't understand why? Can you help me answer this question, please?

Thank you so much Dr Jason Brownlee.

Best,
Dieu Do

Jason Brownlee October 14, 2018 at 6:02 am

This is a common question that I answer l
https://machinelearningmastery.com/faq/single-faq
actual-time-series

Yingfei October 22, 2018 at 1:21 am #

Hi Jason, thanks for your sharing.

I am trying to use AR model to predict a complex-value
the tempreature data in your example code:

series = Series([1, 1+1j, 2, 3, 4, 5, 8, 1+2j, 3, 5])

However, it reports an error message like this:

/anaconda/lib/python3.6/site-packages/statsmodels/tsa/tsatools.py in lagmat(x, maxlag, trim, original, use_pandas)
377 dropidx = nvar
378 if maxlag >= nobs:
–> 379 raise ValueError("maxlag should be < nobs")
380 lm = np.zeros((nobs + maxlag, nvar * (maxlag + 1)))
381 for k in range(0, int(maxlag + 1)):

ValueError: maxlag should be < nobs

Please shed a light on how to correct it.

Thanks.

Jason Brownlee October 22, 2018 at 6:20 am #

REPLY

The error suggests you may need to char

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* awesome at applied machine learning.

Read More

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

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


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


Difference Between Classification and Regression in Machine Learning
DECEMBER 11, 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

Your Start in Machine Learning