



Want help with time series? [Take the FREE Mini-Course](#)

Search...



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

by Jason Brownlee on [February 20, 2017](#) in [Time Series](#)

Tweet

Share

Share

G+

Time series forecasting is a process, and the only way to get good forecasts is to practice this process.

In this tutorial, you will discover how to forecast the monthly sales of French champagne with Python.

Working through this tutorial will provide you with a framework for the steps and the tools for working through your own time series forecasting problems.

After completing this tutorial, you will know:

- How to confirm your Python environment and carefully define a time series forecasting problem.
- How to create a test harness for evaluating models, develop a baseline forecast, and better understand your problem with the tools of time series analysis.
- How to develop an autoregressive integrated moving average model, save it to file, and later load it to make predictions for new time steps.

Let's get started.

- **Update Mar/2017:** Fixed a typo in the code example in the “Review Residual Errors” section where the wrong model was run.

Your Start in Machine Learning



Time Series Forecast Study with Python
Photo by Basheer Tome

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Overview

In this tutorial, we will work through a time series forecasting project from end-to-end, from downloading the dataset and defining the problem to training a final model and making predictions.

This project is not exhaustive, but shows how you can get good results quickly by working through a time series forecasting problem systematically.

The steps of this project that we will through are as follows.

1. Environment.
2. Problem Description.
3. Test Harness.
4. Persistence.
5. Data Analysis.
6. ARIMA Models.
7. Model Validation.

This will provide a template for working through a time series prediction problem that you can use on your own dataset.

Your Start in Machine Learning

Stop learning Time Series Forecasting the *slow way!*

Take my free 7-day email course and discover data prep, modeling and more (with sample code).

Click to sign-up and also get a free PDF Ebook version of the course.

[Start Your FREE Mini-Course Now!](#)

1. Environment

This tutorial assumes an installed and working SciPy environment.

- SciPy
- NumPy
- Matplotlib
- Pandas
- scikit-learn
- statsmodels

If you need help installing Python and the SciPy environment, see my [Python distribution](#) that manages much of it for you.

This script will help you check your installed versions of these libraries:

```

1 # scipy
2 import scipy
3 print('scipy: %s' % scipy.__version__)
4 # numpy
5 import numpy
6 print('numpy: %s' % numpy.__version__)
7 # matplotlib
8 import matplotlib
9 print('matplotlib: %s' % matplotlib.__version__)
10 # pandas
11 import pandas
12 print('pandas: %s' % pandas.__version__)
13 # scikit-learn
14 import sklearn
15 print('sklearn: %s' % sklearn.__version__)
16 # statsmodels
17 import statsmodels
18 print('statsmodels: %s' % statsmodels.__version__)

```

The results on my workstation used to write this tutorial are as follows:

```

1 scipy: 0.18.1
2 numpy: 1.11.2
3 matplotlib: 1.5.3
4 pandas: 0.19.1
5 sklearn: 0.18.1
6 statsmodels: 0.6.1

```

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

Email Address

[START MY EMAIL COURSE](#)

da

Your Start in Machine Learning

2. Problem Description

The problem is to predict the number of monthly sales of champagne for the Perrin Freres label (named for a region in France).

The dataset provides the number of monthly sales of champagne from January 1964 to September 1972, or just under 10 years of data.

The values are a count of millions of sales and there are 105 observations.

The dataset is credited to Makridakis and Wheelwright, 1989.

You can learn more about this dataset and download it [here](#).

Download the dataset as a CSV file and place it in your working directory as `"champagne.csv"`.

3. Test Harness

We must develop a test harness to investigate the data.

This involves two steps:

1. Defining a Validation Dataset.
2. Developing a Method for Model Evaluation.

3.1 Validation Dataset

The dataset is not current. This means that we cannot easily collect updated data to validate the model.

Therefore we will pretend that it is September 1971 and withhold the last one year of data from analysis and model selection.

This final year of data will be used to validate the final model.

The code below will load the dataset as a Pandas Series and split into two, one for model development (`dataset.csv`) and the other for validation (`validation.csv`).

```

1 from pandas import Series
2 series = Series.from_csv('champagne.csv', header=0)
3 split_point = len(series) - 12
4 dataset, validation = series[0:split_point], series[split_point:]
5 print('Dataset %d, Validation %d' % (len(dataset), len(validation)))
6 dataset.to_csv('dataset.csv')
7 validation.to_csv('validation.csv')

```

Running the example creates two files and prints the number of observations in each.

1 Dataset 93, Validation 12

Your Start in Machine Learning

The specific contents of these files are:

- *dataset.csv*: Observations from January 1964 to September 1971 (93 observations)
- *validation.csv*: Observations from October 1971 to September 1972 (12 observations)

The validation dataset is about 11% of the original dataset.

Note that the saved datasets do not have a header line, therefore we do not need to cater for this when working with these files later.

3.2. Model Evaluation

Model evaluation will only be performed on the data in *dataset.csv* generated in the previous section.

Model evaluation involves two elements:

1. Performance Measure.
2. Test Strategy.

3.2.1 Performance Measure

The observations are a count of champagne sales in thousands per month.

We will evaluate the performance of predictions using a metric called Root Mean Squared Error (RMSE). It gives more weight to predictions that are grossly wrong and less weight to predictions that are close to the actual value.

Any transforms to the data must be reversed before the RMSE is calculated so that the scores are comparable. This allows us to directly compare the performance between different methods.

We can calculate the RMSE using the helper function from the scikit-learn library *mean_squared_error()* that calculates the mean squared error between a list of expected values (the test set) and the list of predictions. We can then take the square root of this value to give us an RMSE score.

For example:

```
1 from sklearn.metrics import mean_squared_error
2 from math import sqrt
3 ...
4 test = ...
5 predictions = ...
6 mse = mean_squared_error(test, predictions)
7 rmse = sqrt(mse)
8 print('RMSE: %.3f' % rmse)
```

3.2.2 Test Strategy

Candidate models will be evaluated using walk-forward validation.

This is because a rolling-forecast type model is required from the problem definition. This is where one-step forecasts are needed given all available data.

Your Start in Machine Learning

The walk-forward validation will work as follows:

- The first 50% of the dataset will be held back to train the model.
- The remaining 50% of the dataset will be iterated and test the model.
- For each step in the test dataset:
 - A model will be trained.
 - A one-step prediction made and the prediction stored for later evaluation.
 - The actual observation from the test dataset will be added to the training dataset for the next iteration.
- The predictions made during the iteration of the test dataset will be evaluated and an RMSE score reported.

Given the small size of the data, we will allow a model prediction.

We can write the code for the test harness using simple

Firstly, we can split the dataset into train and test sets dataset to `float32` in case the loaded data still has some

```
1 # prepare data
2 X = series.values
3 X = X.astype('float32')
4 train_size = int(len(X) * 0.50)
5 train, test = X[0:train_size], X[train_size:]
```

Next, we can iterate over the time steps in the test data we need to easily append a new observation each iteration overkill.

The prediction made by the model is called `yhat` for convention, as the outcome or observation is referred to as `y` and `yhat` (a '`y`' with a mark above) is the mathematical notation for the prediction of the `y` variable.

The prediction and observation are printed each observation for a sanity check prediction in case there are issues with the model.

```
1 # walk-forward validation
2 history = [x for x in train]
3 predictions = []
4 for i in range(len(test)):
5     # predict
6     yhat = ...
7     predictions.append(yhat)
8     # observation
9     obs = test[i]
10    history.append(obs)
11    print('>Predicted=%f, Expected=%f' % (yhat, obs))
```

4. Persistence

Your Start in Machine Learning

The first step before getting bogged down in data analysis and modeling is to establish a baseline of performance.

This will provide both a template for evaluating models using the proposed test harness and a performance measure by which all more elaborate predictive models can be compared.

The baseline prediction for time series forecasting is called the naive forecast, or persistence.

This is where the observation from the previous time step is used as the prediction for the observation at the next time step.

We can plug this directly into the test harness defined in the previous section.

The complete code listing is provided below.

```

1  from pandas import Series
2  from sklearn.metrics import mean_squared_error
3  from math import sqrt
4  # load data
5  series = Series.from_csv('dataset.csv')
6  # prepare data
7  X = series.values
8  X = X.astype('float32')
9  train_size = int(len(X) * 0.50)
10 train, test = X[0:train_size], X[train_size:]
11 # walk-forward validation
12 history = [x for x in train]
13 predictions = list()
14 for i in range(len(test)):
15     # predict
16     yhat = history[-1]
17     predictions.append(yhat)
18     # observation
19     obs = test[i]
20     history.append(obs)
21     print('>Predicted=%f, Expected=%f' % (yhat, obs))
22 # report performance
23 mse = mean_squared_error(test, predictions)
24 rmse = sqrt(mse)
25 print('RMSE: %f' % rmse)

```

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Running the test harness prints the prediction and observation for each iteration of the test dataset.

The example ends by printing the RMSE for the model.

In this case, we can see that the persistence model achieved an RMSE of 3186.501. This means that on average, the model was wrong by about 3,186 million sales for each prediction made.

```

1 ...
2 >Predicted=4676.000, Expected=5010
3 >Predicted=5010.000, Expected=4874
4 >Predicted=4874.000, Expected=4633
5 >Predicted=4633.000, Expected=1659
6 >Predicted=1659.000, Expected=5951
7 RMSE: 3186.501

```

We now have a baseline prediction method and perfor

Your Start in Machine Learning

5. Data Analysis

We can use summary statistics and plots of the data to quickly learn more about the structure of the prediction problem.

In this section, we will look at the data from five perspectives:

1. Summary Statistics.
2. Line Plot.
3. Seasonal Line Plots
4. Density Plots.
5. Box and Whisker Plot.

5.1 Summary Statistics

Summary statistics provide a quick look at the limits of what we are working with.

The example below calculates and prints summary sta

```
1 from pandas import Series
2 series = Series.from_csv('dataset.csv')
3 print(series.describe())
```

Running the example provides a number of summary

Some observations from these statistics include:

- The number of observations (count) matches our expectation, meaning we are handling the data correctly.
- The mean is about 4,641, which we might consider our level in this series.
- The standard deviation (average spread from the mean) is relatively large at 2,486 sales.
- The percentiles along with the standard deviation do suggest a large spread to the data.

1	count	93.000000
2	mean	4641.118280
3	std	2486.403841
4	min	1573.000000
5	25%	3036.000000
6	50%	4016.000000
7	75%	5048.000000
8	max	13916.000000

5.2 Line Plot

A line plot of a time series can provide a lot of insight into the problem.

The example below creates and shows a line plot of the dataset.

```
1 from pandas import Series
2 from matplotlib import pyplot
```

Your Start in Machine Learning

You can master applied Machine Learning without the math or fancy degree.

Find out how in this *free* and *practical* email course.

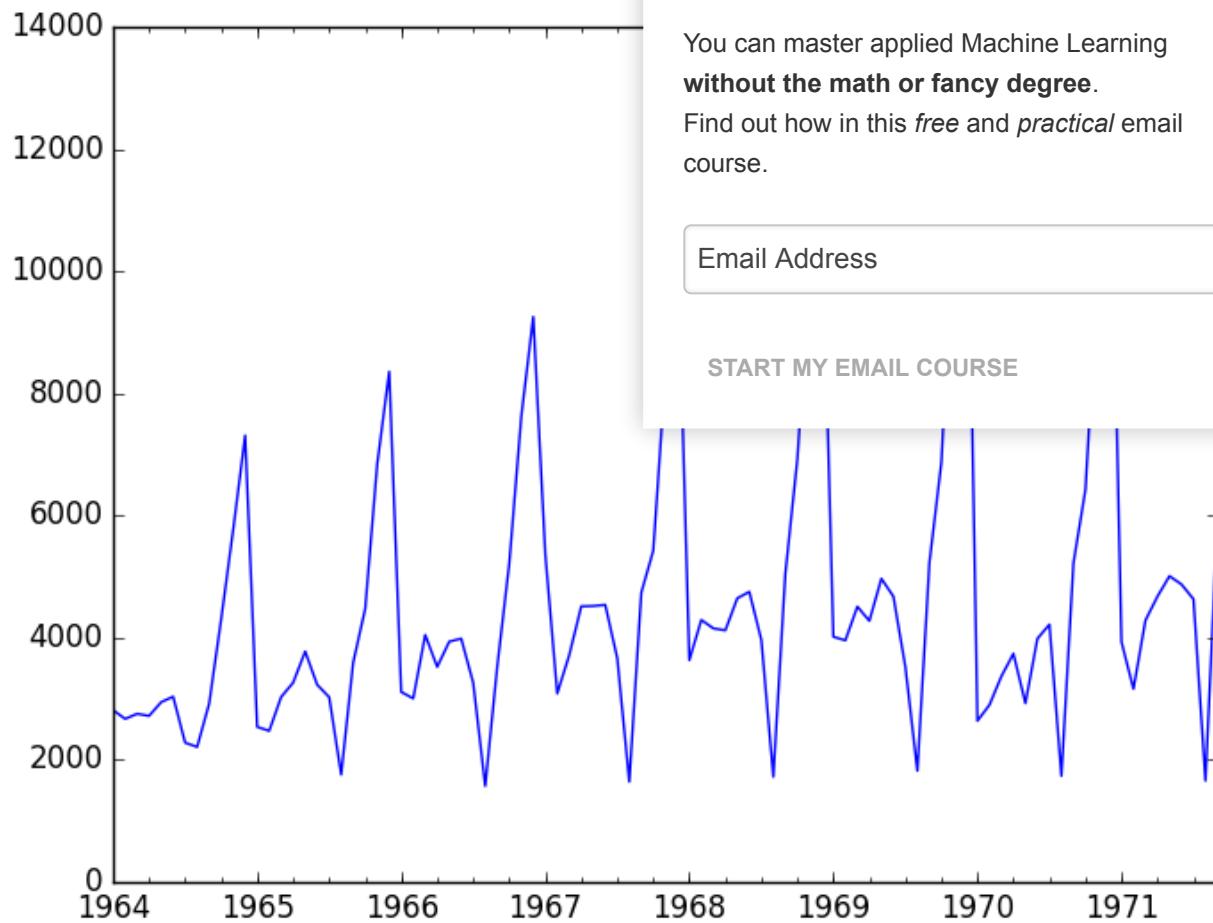
[START MY EMAIL COURSE](#)

```
3 series = Series.from_csv('dataset.csv')
4 series.plot()
5 pyplot.show()
```

Run the example and review the plot. Note any obvious temporal structures in the series.

Some observations from the plot include:

- There may be an increasing trend of sales over time.
- There appears to be systematic seasonality to the sales for each year.
- The seasonal signal appears to be growing over time, suggesting a multiplicative relationship (increasing change).
- There do not appear to be any obvious outliers.
- The seasonality suggests that the series is almost



Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

There may be benefit in explicitly modeling the seasonal component and removing it. You may also explore using differencing with one or two levels in order to make the series stationary.

The increasing trend or growth in the seasonal component may suggest the use of a log or other power transform.

Your Start in Machine Learning

5.3 Seasonal Line Plots

We can confirm the assumption that the seasonality is a yearly cycle by eyeballing line plots of the dataset by year.

The example below takes the 7 full years of data as separate groups and creates one line plot for each. The line plots are aligned vertically to help spot any year-to-year pattern.

```
1 from pandas import Series
2 from pandas import DataFrame
3 from pandas import TimeGrouper
4 from matplotlib import pyplot
5 series = Series.from_csv('dataset.csv')
6 groups = series['1964':'1970'].groupby(TimeGrouper('A'))
7 years = DataFrame()
8 pyplot.figure()
9 i = 1
10 n_groups = len(groups)
11 for name, group in groups:
12     pyplot.subplot((n_groups*100) + 10 + i)
13     i += 1
14     pyplot.plot(group)
15 pyplot.show()
```

Running the example creates the stack of 7 line plots.

We can clearly see a dip each August and a rise from same each year, although at different levels.

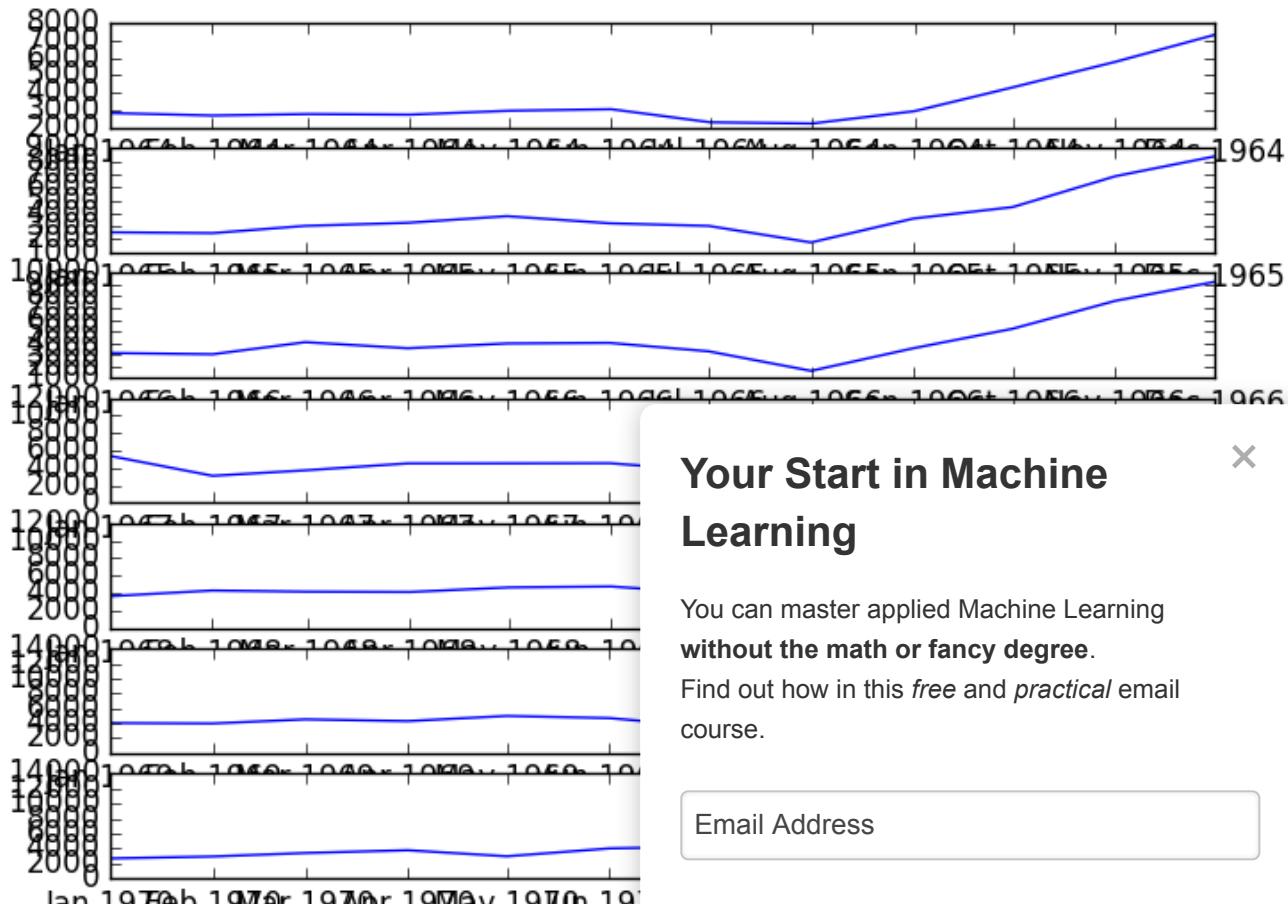
This will help with any explicitly season-based modelin

Your Start in Machine Learning

You can master applied Machine Learning without the math or fancy degree.

Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)



Seasonal Per Year Line Plots

It might have been easier if all season line plots were added to the one graph to help contrast the data for each year.

5.4 Density Plot

Reviewing plots of the density of observations can provide further insight into the structure of the data.

The example below creates a histogram and density plot of the observations without any temporal structure.

```

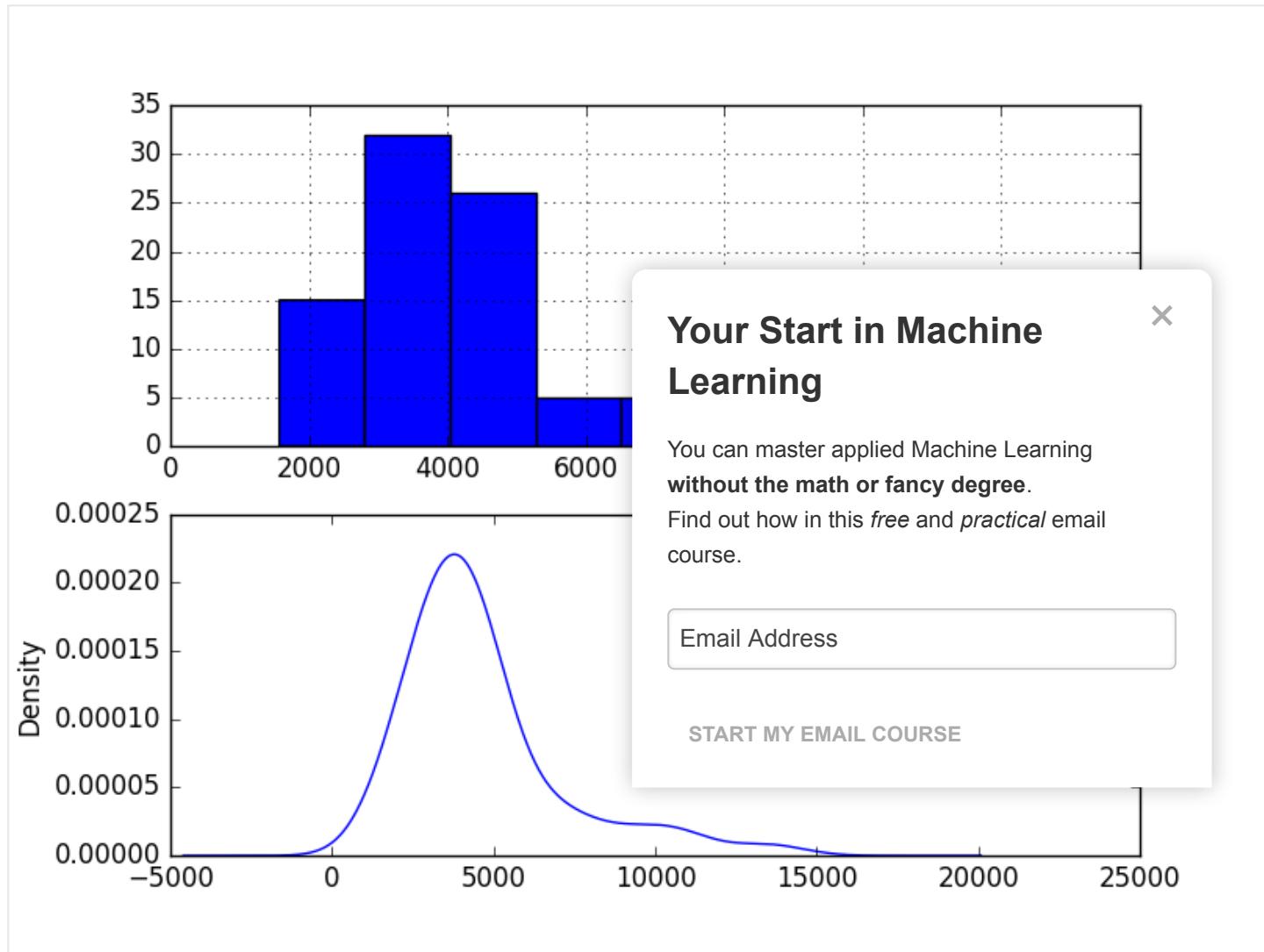
1 from pandas import Series
2 from matplotlib import pyplot
3 series = Series.from_csv('dataset.csv')
4 pyplot.figure(1)
5 pyplot.subplot(211)
6 series.hist()
7 pyplot.subplot(212)
8 series.plot(kind='kde')
9 pyplot.show()
```

Run the example and review the plots.

Your Start in Machine Learning

Some observations from the plots include:

- The distribution is not Gaussian.
- The shape has a long right tail and may suggest an exponential distribution



This lends more support to exploring some power transforms of the data prior to modeling.

5.5 Box and Whisker Plots

We can group the monthly data by year and get an idea of the spread of observations for each year and how this may be changing.

We do expect to see some trend (increasing mean or median), but it may be interesting to see how the rest of the distribution may be changing.

The example below groups the observations by year and creates one box and whisker plot for each year of observations. The last year (1971) only contains 9 months and may not be a useful comparison with the 12 months of observations for other years. Therefore, only data between 1964 and 1970 was plotted.

```
1 from pandas import Series
2 from pandas import DataFrame
```

Your Start in Machine Learning

```

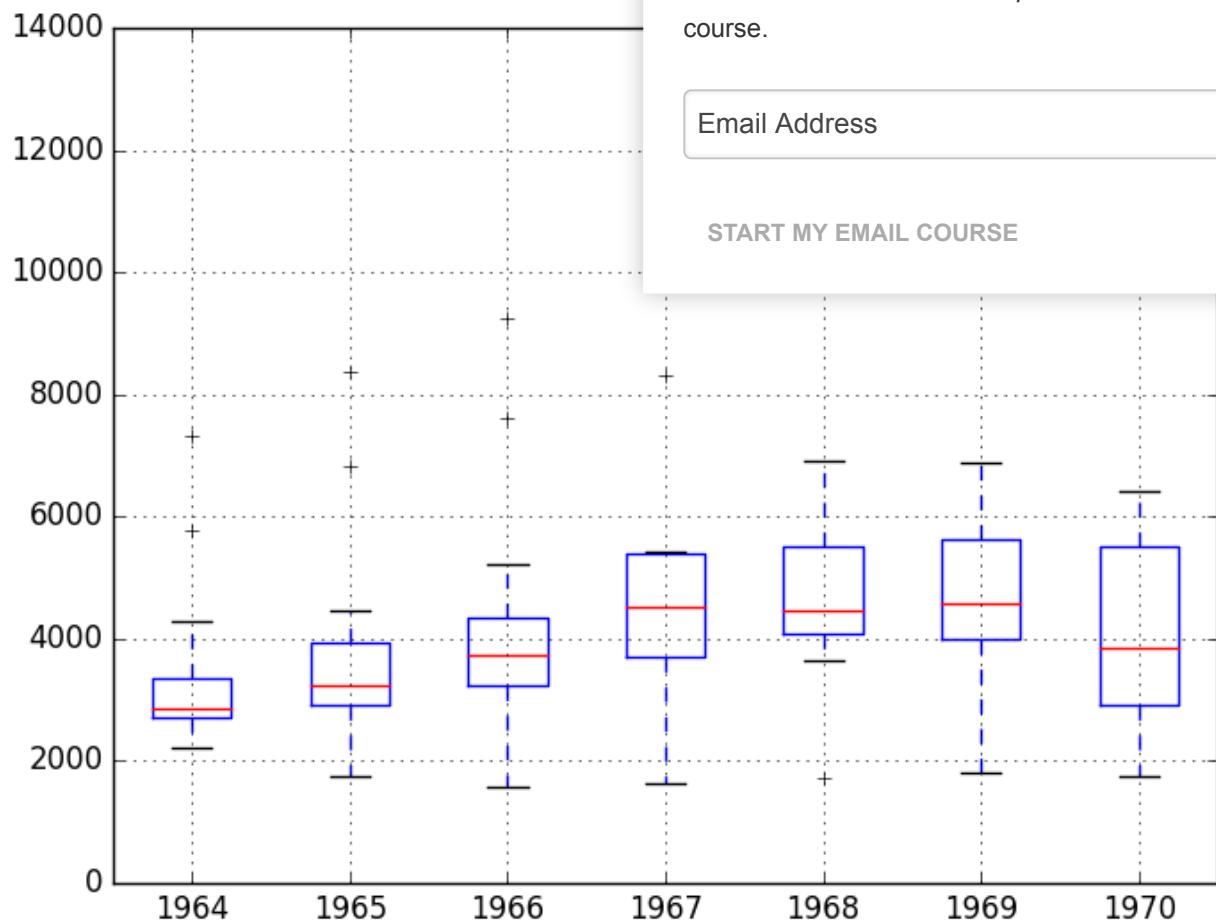
3 from pandas import TimeGrouper
4 from matplotlib import pyplot
5 series = Series.from_csv('dataset.csv')
6 groups = series['1964':'1970'].groupby(TimeGrouper('A'))
7 years = DataFrame()
8 for name, group in groups:
9     years[name.year] = group.values
10 years.boxplot()
11 pyplot.show()

```

Running the example creates 7 box and whisker plots side-by-side, one for each of the 7 years of selected data.

Some observations from reviewing the plots include:

- The median values for each year (red line) may suggest growth over time.
- The spread or middle 50% of the data (blue boxes) also suggests growth over time.
- There are outliers each year (black crosses); these are relatively consistent across the years.
- The last year, 1970, does look different from the others.



Champagne Sales Box and Whisker Plots

The observations suggest perhaps some growth trend over time, but no clear seasonal cycle.

Your Start in Machine Learning

This yearly view of the data is an interesting avenue and could be pursued further by looking at summary statistics from year-to-year and changes in summary stats from year-to-year.

6. ARIMA Models

In this section, we will develop Autoregressive Integrated Moving Average, or ARIMA, models for the problem.

We will approach modeling by both manual and automatic configuration of the ARIMA model. This will be followed by a third step of investigating the residual errors of the chosen model.

As such, this section is broken down into 3 steps:

1. Manually Configure the ARIMA.
2. Automatically Configure the ARIMA.
3. Review Residual Errors.

6.1 Manually Configured ARIMA

The ARIMA(p, d, q) model requires three parameters p , d , and q .

Analysis of the time series data assumes that we are dealing with a stationary process.

The time series is almost certainly non-stationary. We can difference the series and using a statistical test to confirm that the resulting series is stationary.

The seasonality in the series is seemingly year-to-year. Seasonal data can be differenced by subtracting the observation from the same time in the previous cycle, in this case the same month in the previous year. This does mean that we will lose the first year of observations as there is no prior year to difference with.

The example below creates a deseasonalized version of the series and saves it to file *stationary.csv*.

```

1 from pandas import Series
2 from statsmodels.tsa.stattools import adfuller
3 from matplotlib import pyplot
4
5 # create a differenced series
6 def difference(dataset, interval=1):
7     diff = list()
8     for i in range(interval, len(dataset)):
9         value = dataset[i] - dataset[i - interval]
10        diff.append(value)
11    return Series(diff)
12
13 series = Series.from_csv('dataset.csv')
14 X = series.values
15 X = X.astype('float32')
16 # difference data
17 months_in_year = 12
18 stationary = difference(X, months_in_year)
19 stationary.index = series.index[months_in_year:]
20 # check if stationary
21 result = adfuller(stationary)

```

Your Start in Machine Learning

You can master applied Machine Learning without the math or fancy degree. Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Your Start in Machine Learning

```

22 print('ADF Statistic: %f' % result[0])
23 print('p-value: %f' % result[1])
24 print('Critical Values:')
25 for key, value in result[4].items():
26     print('\t%s: %.3f' % (key, value))
27 # save
28 stationary.to_csv('stationary.csv')
29 # plot
30 stationary.plot()
31 pyplot.show()

```

Running the example outputs the result of a statistical significance test of whether the differenced series is stationary. Specifically, the augmented Dickey-Fuller test.

The results show that the test statistic value -7.134898 is smaller than the critical value at 1% of -3.515. This suggests that we can reject the null hypothesis with a low probability that the result is a statistical fluke).

Rejecting the null hypothesis means that the process is not stationary or does not have time-dependent structure.

```

1 ADF Statistic: -7.134898
2 p-value: 0.000000
3 Critical Values:
4      5%: -2.898
5      1%: -3.515
6     10%: -2.586

```

For reference, the seasonal difference operation can be applied to the monthly sales data by subtracting the value from one month the year before. This is needed in the case that there is a seasonal pattern in the data. We will invert the seasonal difference to get back to the original differenced data. The function to invert the seasonal difference is:

```

1 # invert differenced value
2 def inverse_difference(history, yhat, interval=1):
3     return yhat + history[-interval]

```

A plot of the differenced dataset is also created.

The plot does not show any obvious seasonality or trend, suggesting the seasonally differenced dataset is a good starting point for modeling.

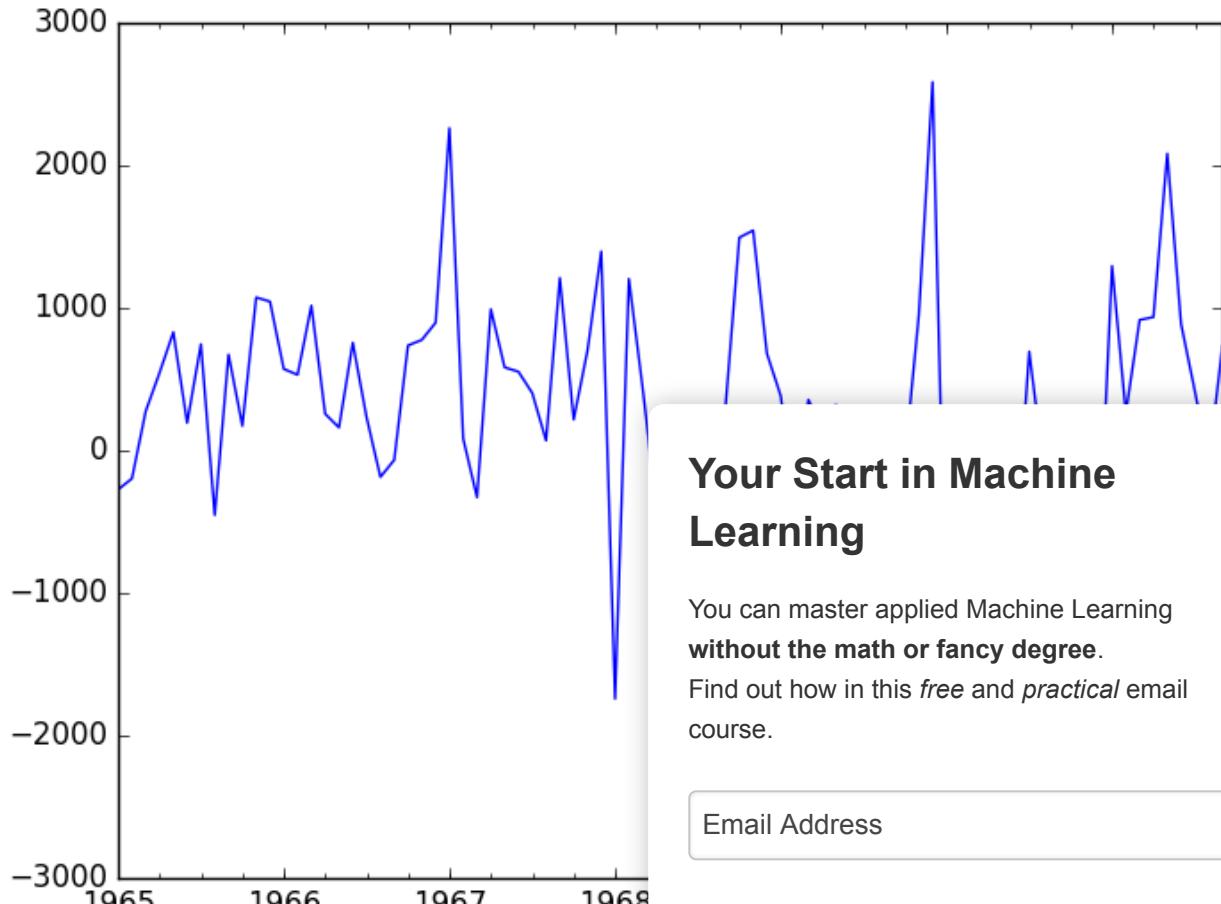
We will use this dataset as an input to the ARIMA model. It also suggests that no further differencing may be required, and that the d parameter may be set to 0.

Your Start in Machine Learning

You can master applied Machine Learning without the math or fancy degree.

Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)



Seasonal Differenced Champagne Sales Line Plot

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)

The next first step is to select the lag values for the Autoregression (AR) and Moving Average (MA) parameters, p and q respectively.

We can do this by reviewing Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots.

Note, we are now using the seasonally differenced stationary.csv as our dataset.

The example below creates ACF and PACF plots for the series.

```

1 from pandas import Series
2 from statsmodels.graphics.tsaplots import plot_acf
3 from statsmodels.graphics.tsaplots import plot_pacf
4 from matplotlib import pyplot
5 series = Series.from_csv('stationary.csv')
6 pyplot.figure()
7 pyplot.subplot(211)
8 plot_acf(series, ax=pyplot.gca())
9 pyplot.subplot(212)
10 plot_pacf(series, ax=pyplot.gca())
11 pyplot.show()
```

Your Start in Machine Learning

Run the example and review the plots for insights into how to set the p and q variables for the ARIMA model.

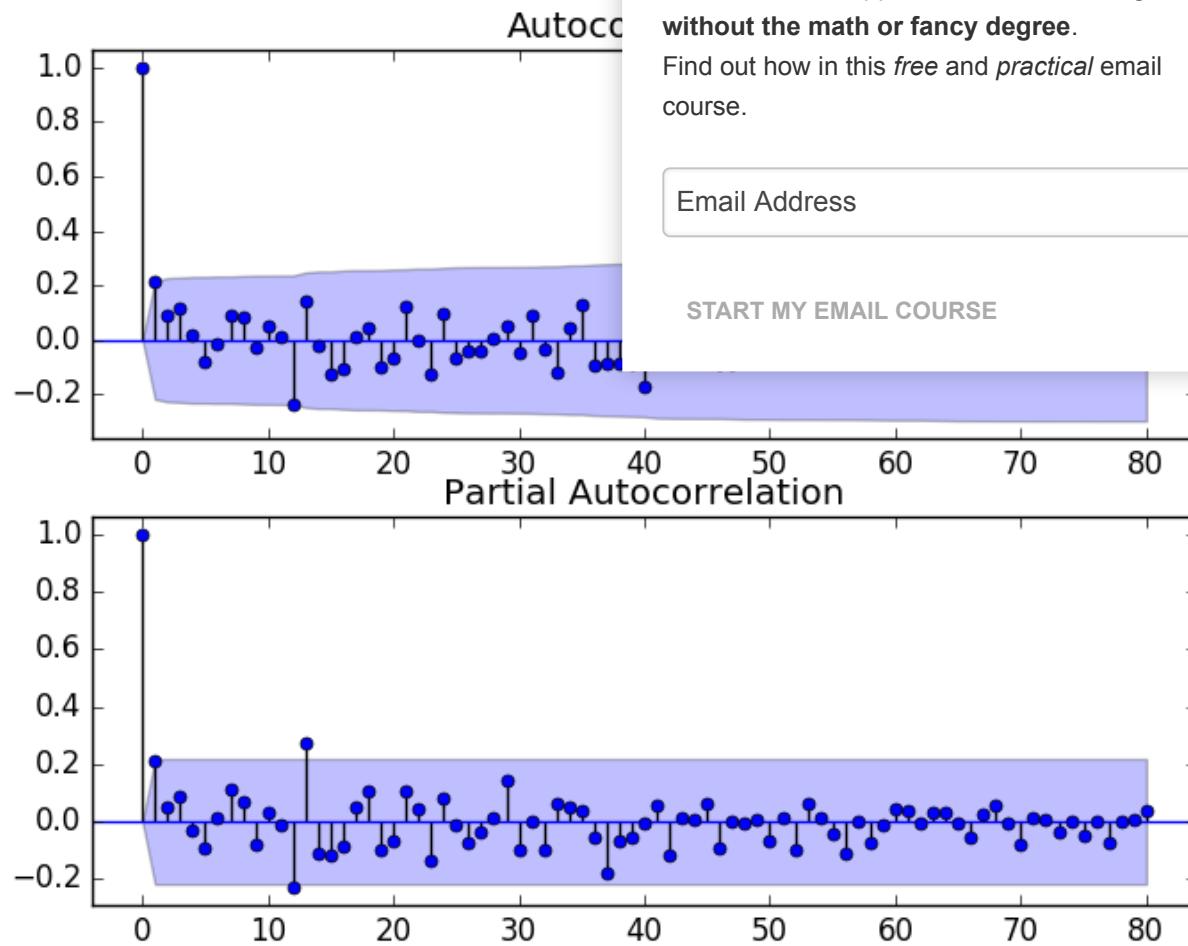
Below are some observations from the plots.

- The ACF shows a significant lag for 1 month.
- The PACF shows a significant lag for 1 month, with perhaps some significant lag at 12 and 13 months.
- Both the ACF and PACF show a drop-off at the same point, perhaps suggesting a mix of AR and MA.

A good starting point for the p and q values is also 1.

The PACF plot also suggests that there is still some seasonality present in the differenced data.

We may consider a better model of seasonality, such as including seasonal differencing in the model rather than seasonal differencing.



Your Start in Machine Learning

You can master applied Machine Learning without the math or fancy degree.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

This quick analysis suggests an ARIMA(1,0,1) on the stationary data may be a good starting point.

Your Start in Machine Learning

The historic observations will be seasonally differenced prior to the fitting of each ARIMA model. The differencing will be inverted for all predictions made to make them directly comparable to the expected observation in the original sale count units.

Experimentation shows that this configuration of ARIMA does not converge and results in errors by the underlying library. Further experimentation showed that adding one level of differencing to the stationary data made the model more stable. The model can be extended to ARIMA(1,1,1).

We will also disable the automatic addition of a trend constant from the model by setting the ‘*trend*’ argument to ‘*nc*’ for no constant in the call to `fit()`. From experimentation, I find that this can result in better forecast performance on some problems.

The example below demonstrates the performance of

```

1  from pandas import Series
2  from sklearn.metrics import mean_squared_error
3  from statsmodels.tsa.arima_model import ARIMA
4  from math import sqrt
5
6  # create a differenced series
7  def difference(dataset, interval=1):
8      diff = list()
9      for i in range(interval, len(dataset)):
10          value = dataset[i] - dataset[i - interval]
11          diff.append(value)
12      return diff
13
14 # invert differenced value
15 def inverse_difference(history, yhat, interval=1):
16     return yhat + history[-interval]
17
18 # load data
19 series = Series.from_csv('dataset.csv')
20 # prepare data
21 X = series.values
22 X = X.astype('float32')
23 train_size = int(len(X) * 0.50)
24 train, test = X[0:train_size], X[train_size:]
25 # walk-forward validation
26 history = [x for x in train]
27 predictions = list()
28 for i in range(len(test)):
29     # difference data
30     months_in_year = 12
31     diff = difference(history, months_in_year)
32     # predict
33     model = ARIMA(diff, order=(1,1,1))
34     model_fit = model.fit(trend='nc', disp=0)
35     yhat = model_fit.forecast()[0]
36     yhat = inverse_difference(history, yhat, months_in_year)
37     predictions.append(yhat)
38     # observation
39     obs = test[i]
40     history.append(obs)
41     print('>Predicted=%f, Expected=%f' % (yhat, obs))
42 # report performance
43 mse = mean_squared_error(test, predictions)
44 rmse = sqrt(mse)
45 print('RMSE: %f' % rmse)
```

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

Email Address

[START MY EMAIL COURSE](#)

Your Start in Machine Learning

Note, you may see a warning message from the underlying linear algebra library; this can be ignored for now.

Running this example results in an RMSE of 956.942, which is dramatically better than the persistence RMSE of 3186.501.

```

1 ...
2 >Predicted=3157.018, Expected=5010
3 >Predicted=4615.082, Expected=4874
4 >Predicted=4624.998, Expected=4633
5 >Predicted=2044.097, Expected=1659
6 >Predicted=5404.428, Expected=5951
7 RMSE: 956.942

```

This is a great start, but we may be able to get improvement.

6.2 Grid Search ARIMA Hyperparameters

The ACF and PACF plots suggest that an ARIMA(1,0,1) model may be appropriate.

To confirm this analysis, we can grid search a suite of hyperparameters and find a combination that results in better out of sample RMSE performance.

In this section, we will search values of p , d , and q for the ARIMA model and find the combination that results in the best performance. We will explore all combinations in a subset of integer values.

Specifically, we will search all combinations of the following hyperparameters:

- p : 0 to 6.
- d : 0 to 2.
- q : 0 to 6.

This is $(7 * 3 * 7)$, or 147, potential runs of the test harness and will take some time to execute.

It may be interesting to evaluate MA models with a lag of 12 or 13 as were noticed as potentially interesting from reviewing the ACF and PACF plots. Experimentation suggested that these models may not be stable, resulting in errors in the underlying mathematical libraries.

The complete worked example with the grid search version of the test harness is listed below.

```

1 import warnings
2 from pandas import Series
3 from statsmodels.tsa.arima_model import ARIMA
4 from sklearn.metrics import mean_squared_error
5 from math import sqrt
6 import numpy
7
8 # create a differenced series
9 def difference(dataset, interval=1):
10     diff = list()
11     for i in range(interval, len(dataset)):
12         value = dataset[i] - dataset[i - inte

```

Your Start in Machine Learning

You can master applied Machine Learning without the math or fancy degree.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Your Start in Machine Learning

```

13     diff.append(value)
14 return numpy.array(diff)
15
16 # invert differenced value
17 def inverse_difference(history, yhat, interval=1):
18     return yhat + history[-interval]
19
20 # evaluate an ARIMA model for a given order (p,d,q) and return RMSE
21 def evaluate_arima_model(X, arima_order):
22     # prepare training dataset
23     X = X.astype('float32')
24     train_size = int(len(X) * 0.50)
25     train, test = X[0:train_size], X[train_size:]
26     history = [x for x in train]
27     # make predictions
28     predictions = list()
29     for t in range(len(test)):
30         # difference data
31         months_in_year = 12
32         diff = difference(history, months_in_year)
33         model = ARIMA(diff, order=arima_order)
34         model_fit = model.fit(trend='nc', disp=0)
35         yhat = model_fit.forecast()[0]
36         yhat = inverse_difference(history, yhat, interval=1)
37         predictions.append(yhat)
38         history.append(test[t])
39     # calculate out of sample error
40     mse = mean_squared_error(test, predictions)
41     rmse = sqrt(mse)
42     return rmse
43
44 # evaluate combinations of p, d and q values
45 def evaluate_models(dataset, p_values, d_values, q_values):
46     dataset = dataset.astype('float32')
47     best_score, best_cfg = float("inf"), None
48     for p in p_values:
49         for d in d_values:
50             for q in q_values:
51                 order = (p,d,q)
52                 try:
53                     mse = evaluate_arima_model(dataset, order)
54                     if mse < best_score:
55                         best_score, best_cfg = mse, order
56                         print('ARIMA%s RMSE=%f' % (order,mse))
57                 except:
58                     continue
59     print('Best ARIMA%s RMSE=%f' % (best_cfg, best_score))
60
61 # load dataset
62 series = Series.from_csv('dataset.csv')
63 # evaluate parameters
64 p_values = range(0, 7)
65 d_values = range(0, 3)
66 q_values = range(0, 7)
67 warnings.filterwarnings("ignore")
68 evaluate_models(series.values, p_values, d_values, q_values)

```

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)

Running the example runs through all combinations and reports the results on those that converge without error. The example takes a little over 2 minutes to run on modern hardware.

The results show that the best configuration discovered was ARIMA(0, 0, 1) with an RMSE of 939.464, slightly lower than the manually configured ARIMA from the previous section.

not be statistically significant.

```

1 ...
2 ARIMA(5, 1, 2) RMSE=1003.200
3 ARIMA(5, 2, 1) RMSE=1053.728
4 ARIMA(6, 0, 0) RMSE=996.466
5 ARIMA(6, 1, 0) RMSE=1018.211
6 ARIMA(6, 1, 1) RMSE=1023.762
7 Best ARIMA(0, 0, 1) RMSE=939.464

```

We will select this ARIMA(0, 0, 1) model going forward.

6.3 Review Residual Errors

A good final check of a model is to review residual forecast errors.

Ideally, the distribution of residual errors should be a normal bell curve.

We can check this by using summary statistics and plotting the residuals of the ARIMA(0, 0, 1) model. The example below calculates the residuals.

```

1 from pandas import Series
2 from pandas import DataFrame
3 from statsmodels.tsa.arima_model import ARIMA
4 from matplotlib import pyplot
5
6 # create a differenced series
7 def difference(dataset, interval=1):
8     diff = list()
9     for i in range(interval, len(dataset)):
10         value = dataset[i] - dataset[i - interval]
11         diff.append(value)
12     return diff
13
14 # invert differenced value
15 def inverse_difference(history, yhat, interval=1):
16     return yhat + history[-interval]
17
18 # load data
19 series = Series.from_csv('dataset.csv')
20 # prepare data
21 X = series.values
22 X = X.astype('float32')
23 train_size = int(len(X) * 0.50)
24 train, test = X[0:train_size], X[train_size:]
25 # walk-forward validation
26 history = [x for x in train]
27 predictions = list()
28 for i in range(len(test)):
29     # difference data
30     months_in_year = 12
31     diff = difference(history, months_in_year)
32     # predict
33     model = ARIMA(diff, order=(0,0,1))
34     model_fit = model.fit(trend='nc', disp=0)
35     yhat = model_fit.forecast()[0]
36     yhat = inverse_difference(history, yhat, months_in_year)
37     predictions.append(yhat)
38     # observation
39     obs = test[i]

```

Your Start in Machine Learning

You can master applied Machine Learning without the math or fancy degree.

Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)

Your Start in Machine Learning

```

40     history.append(obs)
41 # errors
42 residuals = [test[i]-predictions[i] for i in range(len(test))]
43 residuals = DataFrame(residuals)
44 print(residuals.describe())
45 # plot
46 pyplot.figure()
47 pyplot.subplot(211)
48 residuals.hist(ax=pyplot.gca())
49 pyplot.subplot(212)
50 residuals.plot(kind='kde', ax=pyplot.gca())
51 pyplot.show()

```

Running the example first describes the distribution of the residuals.

We can see that the distribution has a right shift and that the mean is non-zero at 165.904728.

This is perhaps a sign that the predictions are biased.

1	count	47.000000
2	mean	165.904728
3	std	934.696199
4	min	-2164.247449
5	25%	-289.651596
6	50%	191.759548
7	75%	732.992187
8	max	2367.304748

The distribution of residual errors is also plotted.

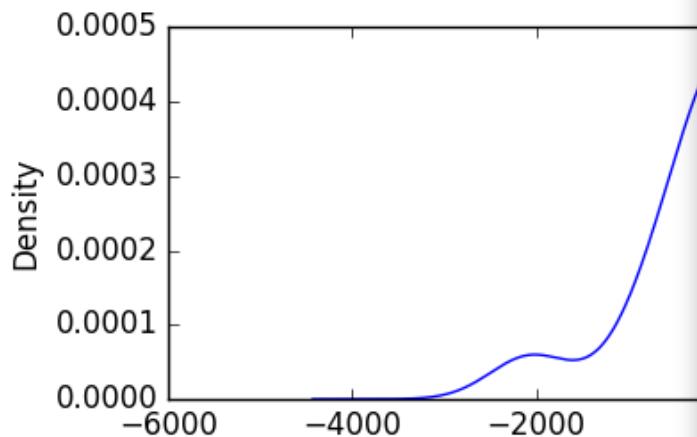
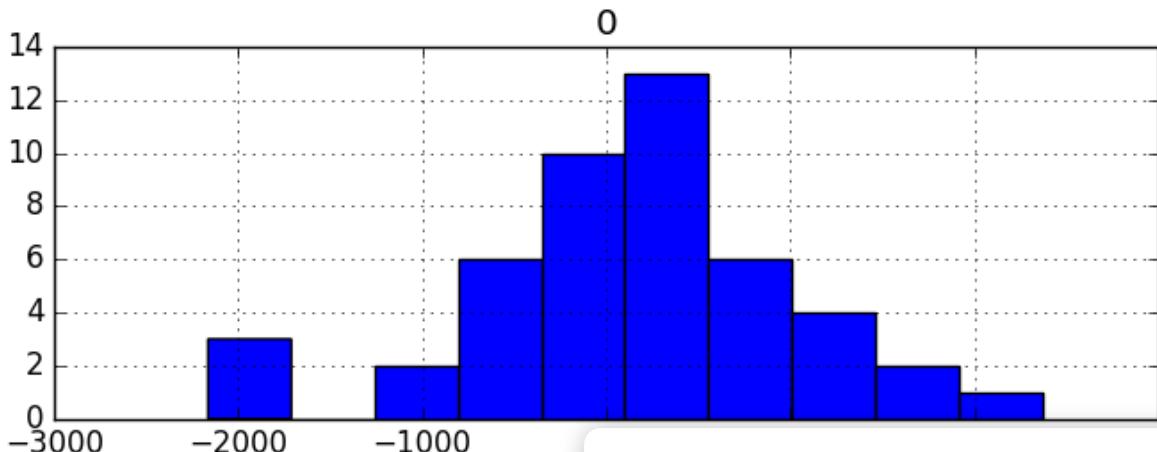
The graphs suggest a Gaussian-like distribution with a right shift.
perhaps a power transform might be worth exploring.

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)

Your Start in Machine Learning



Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**. Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)

Residual Forecast Error Density Plots

We could use this information to bias-correct predictions by adding the mean residual error of 165.904728 to each forecast made.

The example below performs this bias correction.

```

1 from pandas import Series
2 from pandas import DataFrame
3 from statsmodels.tsa.arima_model import ARIMA
4 from matplotlib import pyplot
5 from sklearn.metrics import mean_squared_error
6 from math import sqrt
7
8 # create a differenced series
9 def difference(dataset, interval=1):
10     diff = list()
11     for i in range(interval, len(dataset)):
12         value = dataset[i] - dataset[i - interval]
13         diff.append(value)
14     return diff
15
16 # invert differenced value
17 def inverse_difference(history, yhat, interval=1):
18     return yhat + history[-interval]
19

```

Your Start in Machine Learning

```

20 # load data
21 series = Series.from_csv('dataset.csv')
22 # prepare data
23 X = series.values
24 X = X.astype('float32')
25 train_size = int(len(X) * 0.50)
26 train, test = X[0:train_size], X[train_size:]
27 # walk-forward validation
28 history = [x for x in train]
29 predictions = list()
30 bias = 165.904728
31 for i in range(len(test)):
32     # difference data
33     months_in_year = 12
34     diff = difference(history, months_in_year)
35     # predict
36     model = ARIMA(diff, order=(0,0,1))
37     model_fit = model.fit(trend='nc', disp=0)
38     yhat = model_fit.forecast()[0]
39     yhat = bias + inverse_difference(history,
40     predictions.append(yhat)
41     # observation
42     obs = test[i]
43     history.append(obs)
44 # report performance
45 mse = mean_squared_error(test, predictions)
46 rmse = sqrt(mse)
47 print('RMSE: %.3f' % rmse)
48 # errors
49 residuals = [test[i]-predictions[i] for i in
50 residuals = DataFrame(residuals)
51 print(residuals.describe())
52 # plot
53 pyplot.figure()
54 pyplot.subplot(211)
55 residuals.hist(ax=pyplot.gca())
56 pyplot.subplot(212)
57 residuals.plot(kind='kde', ax=pyplot.gca())
58 pyplot.show()

```

Your Start in Machine Learning

You can master applied Machine Learning without the math or fancy degree.

Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)

The performance of the predictions is improved very slightly from 939.464 to 924.699, which may or may not be significant.

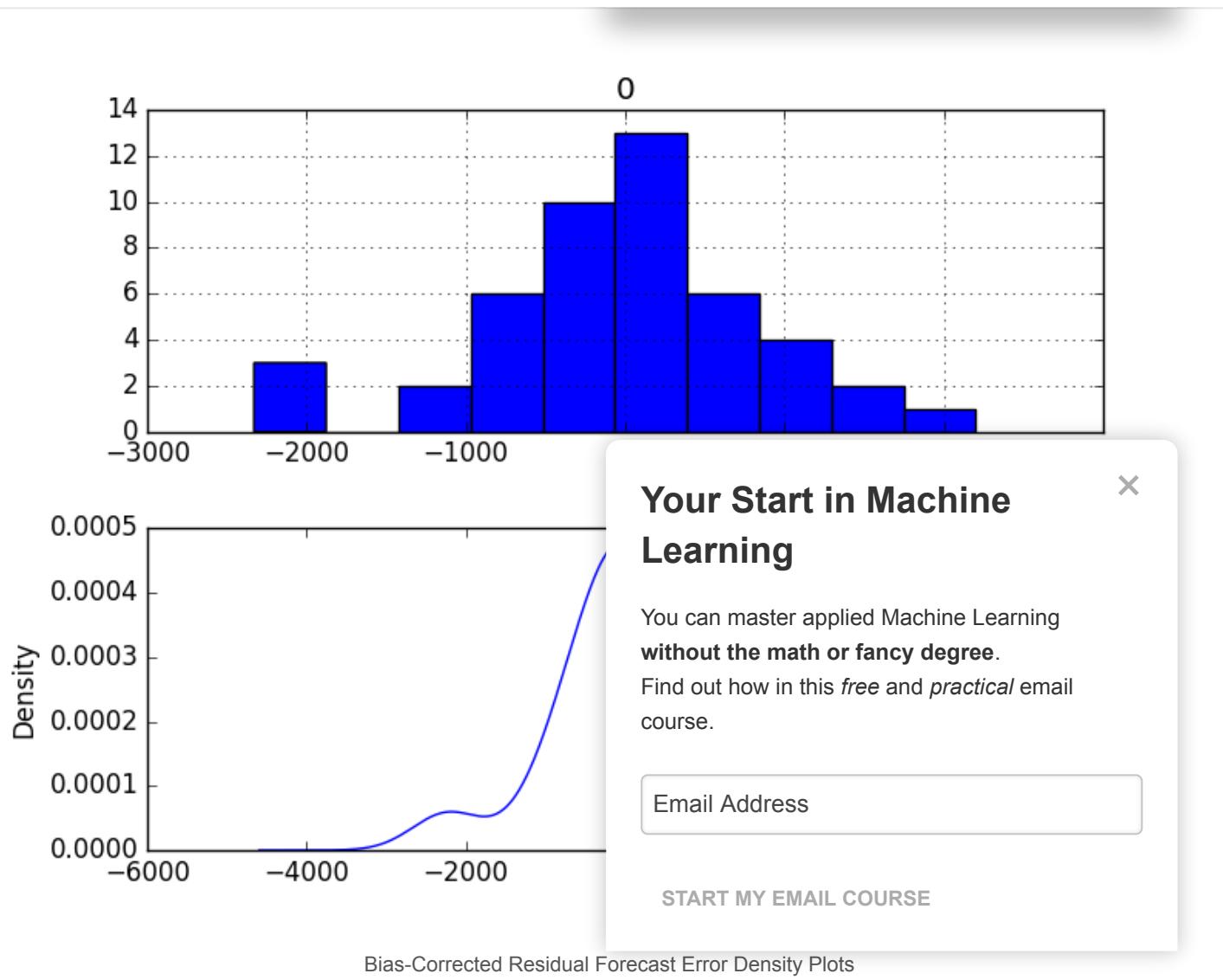
The summary of the forecast residual errors shows that the mean was indeed moved to a value very close to zero.

1	RMSE: 924.699
2	
3	count 4.700000e+01
4	mean 4.965016e-07
5	std 9.346962e+02
6	min -2.330152e+03
7	25% -4.555563e+02
8	50% 2.585482e+01
9	75% 5.670875e+02
10	max 2.201400e+03

Finally, density plots of the residual error do show a small shift towards zero.

It is debatable whether this bias correction is worth it, but we will use it for now.

Your Start in Machine Learning



It is also a good idea to check the time series of the residual errors for any type of autocorrelation. If present, it would suggest that the model has more opportunity to model the temporal structure in the data.

The example below re-calculates the residual errors and creates ACF and PACF plots to check for any significant autocorrelation.

```

1 from pandas import Series
2 from pandas import DataFrame
3 from statsmodels.tsa.arima_model import ARIMA
4 from matplotlib import pyplot
5 from statsmodels.graphics.tsaplots import plot_acf
6 from statsmodels.graphics.tsaplots import plot_pacf
7
8 # create a differenced series
9 def difference(dataset, interval=1):
10     diff = list()
11     for i in range(interval, len(dataset)):
12         value = dataset[i] - dataset[i - interval]
13         diff.append(value)
14     return diff
15
16 # invert differenced value
17 def inverse_difference(history, yhat, interval):
18     return yhat + history[-interval]

```

Your Start in Machine Learning

```

19
20 # load data
21 series = Series.from_csv('dataset.csv')
22 # prepare data
23 X = series.values
24 X = X.astype('float32')
25 train_size = int(len(X) * 0.50)
26 train, test = X[0:train_size], X[train_size:]
27 # walk-forward validation
28 history = [x for x in train]
29 predictions = list()
30 for i in range(len(test)):
31     # difference data
32     months_in_year = 12
33     diff = difference(history, months_in_year)
34     # predict
35     model = ARIMA(diff, order=(0,0,1))
36     model_fit = model.fit(trend='nc', disp=0)
37     yhat = model_fit.forecast()[0]
38     yhat = inverse_difference(history, yhat,
39     predictions.append(yhat)
40     # observation
41     obs = test[i]
42     history.append(obs)
43 # errors
44 residuals = [test[i]-predictions[i] for i in
45 residuals = DataFrame(residuals)
46 print(residuals.describe())
47 # plot
48 pyplot.figure()
49 pyplot.subplot(211)
50 plot_acf(residuals, ax=pyplot.gca())
51 pyplot.subplot(212)
52 plot_pacf(residuals, ax=pyplot.gca())
53 pyplot.show()

```

The results suggest that what little autocorrelation is present in the time series has been captured by the model.

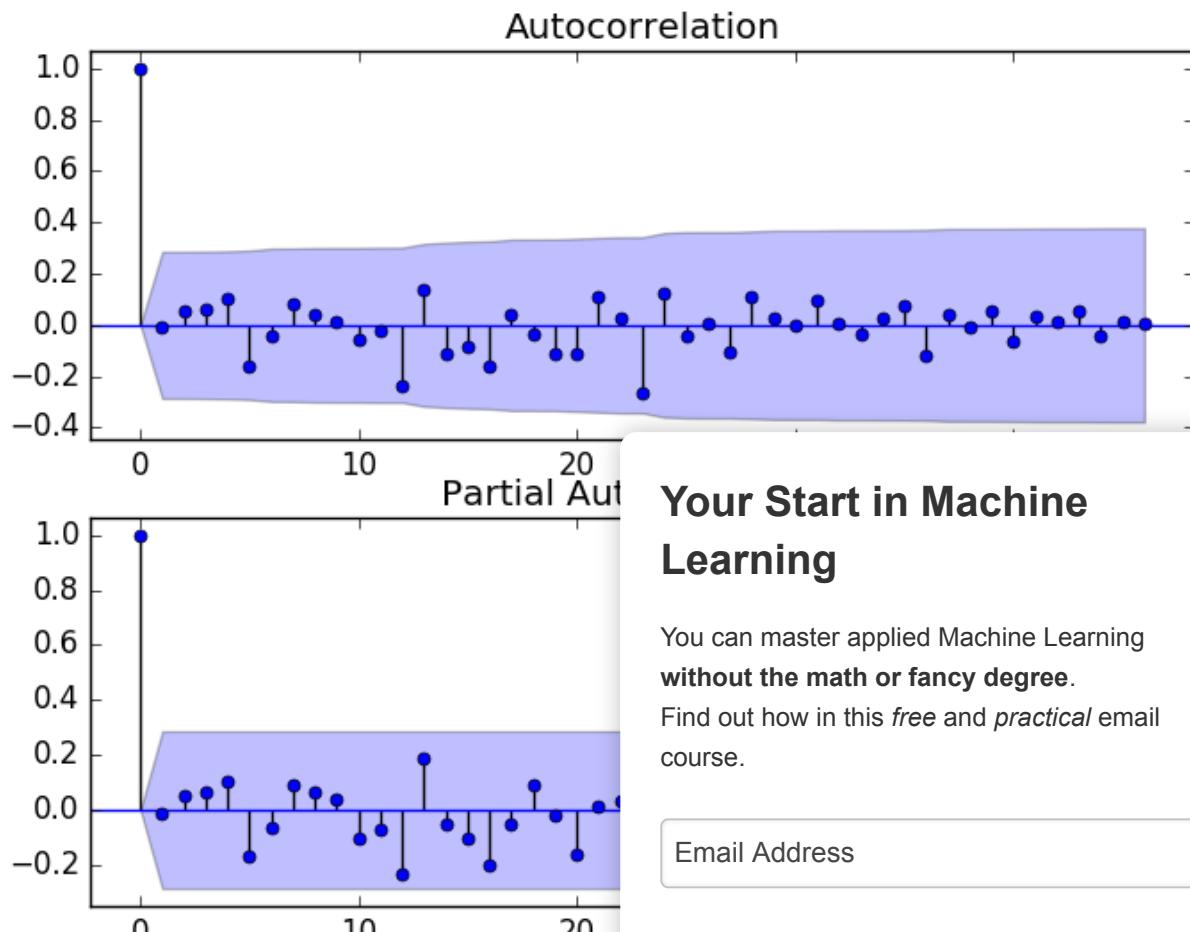
Your Start in Machine Learning

You can master applied Machine Learning without the math or fancy degree.

Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)

Your Start in Machine Learning



Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**. Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)

Residual Forecast Error ACF and PACF Plots

7. Model Validation

After models have been developed and a final model selected, it must be validated and finalized.

Validation is an optional part of the process, but one that provides a ‘last check’ to ensure we have not fooled or misled ourselves.

This section includes the following steps:

1. **Finalize Model:** Train and save the final model.
2. **Make Prediction:** Load the finalized model and make a prediction.
3. **Validate Model:** Load and validate the final model.

7.1 Finalize Model

Finalizing the model involves fitting an ARIMA model on the entire dataset, in this case on a transformed version of the entire dataset.

Once fit, the model can be saved to file for later use.

Your Start in Machine Learning

The example below trains an ARIMA(0,0,1) model on the dataset and saves the whole fit object and the bias to file.

There is a bug in the current stable version of the statsmodels library (v0.6.1) that results in an error when you try to load a saved ARIMA model from file. The error reported is:

```
1 TypeError: __new__() takes at least 3 arguments (1 given)
```

This bug also seems present in the 0.8 release candidate 1 of statsmodels when I tested it. For more details, see [Zae Myung Kim's discussion and fix in this GitHub issue](#).

We can work around this with a monkey patch that adds a `__getnewargs__()` instance function to the ARIMA class before saving.

The example below saves the fit model to file in the code editor.

```
1 from pandas import Series
2 from statsmodels.tsa.arima_model import ARIMA
3 from scipy.stats import boxcox
4 import numpy
5
6 # monkey patch around bug in ARIMA class
7 def __getnewargs__(self):
8     return ((self.endog),(self.k_lags, self.k_ar,
9             self.k_ma), self.exog_names)
10 ARIMA.__getnewargs__ = __getnewargs__
11
12 # create a differenced series
13 def difference(dataset, interval=1):
14     diff = list()
15     for i in range(interval, len(dataset)):
16         value = dataset[i] - dataset[i - interval]
17         diff.append(value)
18     return diff
19
20 # load data
21 series = Series.from_csv('dataset.csv')
22 # prepare data
23 X = series.values
24 X = X.astype('float32')
25 # difference data
26 months_in_year = 12
27 diff = difference(X, months_in_year)
28 # fit model
29 model = ARIMA(diff, order=(0,0,1))
30 model_fit = model.fit(trend='nc', disp=0)
31 # bias constant, could be calculated from in-sample mean residual
32 bias = 165.904728
33 # save model
34 model_fit.save('model.pkl')
35 numpy.save('model_bias.npy', [bias])
```

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)

Running the example creates two local files:

- `model.pkl` This is the ARIMAResult object from the call to `ARIMA.fit()`. This includes the coefficients and all other internal data returned when fitting the model.
- `model_bias.npy` This is the bias value stored as a

Your Start in Machine Learning

7.2 Make Prediction

A natural case may be to load the model and make a single forecast.

This is relatively straightforward and involves restoring the saved model and the bias and calling the `forecast()` method. To invert the seasonal differencing, the historical data must also be loaded.

The example below loads the model, makes a prediction for the next time step, and prints the prediction.

```

1 from pandas import Series
2 from statsmodels.tsa.arima_model import ARIMAResults
3 import numpy
4
5 # invert differenced value
6 def inverse_difference(history, yhat, interval):
7     return yhat + history[-interval]
8
9 series = Series.from_csv('dataset.csv')
10 months_in_year = 12
11 model_fit = ARIMAResults.load('model.pkl')
12 bias = numpy.load('model_bias.npy')
13 yhat = float(model_fit.forecast()[0])
14 yhat = bias + inverse_difference(series.values, 1)
15 print('Predicted: %.3f' % yhat)
```

Running the example prints the prediction of about 6794.773.

```
1 Predicted: 6794.773
```

If we peek inside `validation.csv`, we can see that the value is 6794.773.

The prediction is in the right ballpark.

7.3 Validate Model

We can load the model and use it in a pretend operational manner.

In the test harness section, we saved the final 12 months of the original dataset in a separate file to validate the final model.

We can load this `validation.csv` file now and use it see how well our model really is on “unseen” data.

There are two ways we might proceed:

- Load the model and use it to forecast the next 12 months. The forecast beyond the first one or two months will quickly start to degrade in skill.
- Load the model and use it in a rolling-forecast manner, updating the transform and model for each time step. This is the preferred method as it is how one would use this model in practice as it would achieve the best performance.

As with model evaluation in previous sections, we will make predictions in a rolling-forecast manner. This means that we will step over lead times in the validation set to

Your Start in Machine Learning

the history.

```

1 from pandas import Series
2 from matplotlib import pyplot
3 from statsmodels.tsa.arima_model import ARIMA
4 from statsmodels.tsa.arima_model import ARIMAResults
5 from sklearn.metrics import mean_squared_error
6 from math import sqrt
7 import numpy
8
9 # create a differenced series
10 def difference(dataset, interval=1):
11     diff = list()
12     for i in range(interval, len(dataset)):
13         value = dataset[i] - dataset[i - interval]
14         diff.append(value)
15     return diff
16
17 # invert differenced value
18 def inverse_difference(history, yhat, interval=1):
19     return yhat + history[-interval]
20
21 # load and prepare datasets
22 dataset = Series.from_csv('dataset.csv')
23 X = dataset.values.astype('float32')
24 history = [x for x in X]
25 months_in_year = 12
26 validation = Series.from_csv('validation.csv')
27 y = validation.values.astype('float32')
28 # load model
29 model_fit = ARIMAResults.load('model.pkl')
30 bias = numpy.load('model_bias.npy')
31 # make first prediction
32 predictions = list()
33 yhat = float(model_fit.forecast()[0])
34 yhat = bias + inverse_difference(history, yhat, months_in_year)
35 predictions.append(yhat)
36 history.append(y[0])
37 print('>Predicted=%f, Expected=%f' % (yhat, y[0]))
38 # rolling forecasts
39 for i in range(1, len(y)):
40     # difference data
41     months_in_year = 12
42     diff = difference(history, months_in_year)
43     # predict
44     model = ARIMA(diff, order=(0,0,1))
45     model_fit = model.fit(trend='nc', disp=0)
46     yhat = model_fit.forecast()[0]
47     yhat = bias + inverse_difference(history, yhat, months_in_year)
48     predictions.append(yhat)
49     # observation
50     obs = y[i]
51     history.append(obs)
52     print('>Predicted=%f, Expected=%f' % (yhat, obs))
53 # report performance
54 mse = mean_squared_error(y, predictions)
55 rmse = sqrt(mse)
56 print('RMSE: %.3f' % rmse)
57 pyplot.plot(y)
58 pyplot.plot(predictions, color='red')
59 pyplot.show()
```

Running the example prints each prediction and expe

Your Start in Machine Learning

You can master applied Machine Learning without the math or fancy degree.

Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)

The final RMSE for the validation period is predicted at 361.110 million sales.

This is much better than the expectation of an error of a little more than 924 million sales per month.

```

1 >Predicted=6794.773, Expected=6981
2 >Predicted=10101.763, Expected=9851
3 >Predicted=13219.067, Expected=12670
4 >Predicted=3996.535, Expected=4348
5 >Predicted=3465.934, Expected=3564
6 >Predicted=4522.683, Expected=4577
7 >Predicted=4901.336, Expected=4788
8 >Predicted=5190.094, Expected=4618
9 >Predicted=4930.190, Expected=5312
10 >Predicted=4944.785, Expected=4298
11 >Predicted=1699.409, Expected=1413
12 >Predicted=6085.324, Expected=5877
13 RMSE: 361.110

```

A plot of the predictions compared to the validation data.

At this scale on the plot, the 12 months of forecast sales

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)



Forecast for Champagne Sales Validation Dataset

Your Start in Machine Learning

Summary

In this tutorial, you discovered the steps and the tools for a time series forecasting project with Python.

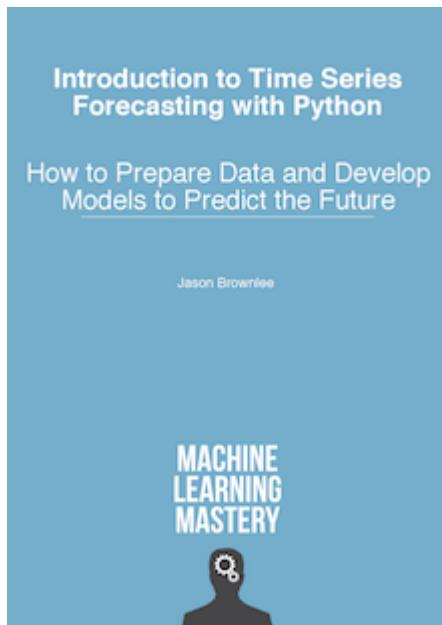
We have covered a lot of ground in this tutorial; specifically:

- How to develop a test harness with a performance measure and evaluation method and how to quickly develop a baseline forecast and skill.
- How to use time series analysis to raise ideas for how to best model the forecast problem.
- How to develop an ARIMA model, save it, and later load it to make predictions on new data.

How did you do? Do you have any questions about this tutorial?

Ask your questions in the comments below and I will do my best to answer them.

Want to Develop Time Series Models?



Develop

Introduc

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

Your Start in Machine Learning

You can master applied Machine Learning without the math or fancy degree.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Finally Bring Time Series Forecasting to Your Own Projects

Skip the Academics. Just Results.

[Click to learn more.](#)

Tweet

Share

Share

G+



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 Álvaro Lemos got a Machine Learning Internship on a Data Science Project

Your Start in Machine Learning

117 Responses to *Time Series Forecast Study with Python: Monthly Sales of French Champagne*



Benson Dube February 20, 2017 at 8:46 pm #

REPLY ↗

Thank You Jason.

Would you recommend this example to a starter. I am a beginner and I am currently learning Python syntax and understand it.

Regards,

Benson



Jason Brownlee February 21, 2017 at 9:34 am #

Hi Benson, this tutorial is an example for a machine learning project. I assume you know your way around Python first (or have some experience).

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Benson March 1, 2017 at 6:45 am #

REPLY ↗

A surely steep learning curve taking you out of your comfort zone, but that's the way to learn. ?



Jason Brownlee March 1, 2017 at 8:48 am #

REPLY ↗

For sure. Dive in!



Viral Mehta February 24, 2017 at 4:56 am #

REPLY ↗

Jason, thanks for very detailed instructions on how to construct the model. When I add a few additional periods in the validation set (manually) for a short-term forecast beyond the dataset, the model won't run until I provide some 'fake' targets (expected y). However, when I provide some fake targets, I see that model quickly adjusts to those targets. I tried different levels of y values and I see model predicted accordingly, shouldn't the predictions be independent of the target values?

Your Start in Machine Learning



Jason Brownlee February 24, 2017 at 10:14 am #

REPLY ↗

You should not need fake targets Viral.

You can forecast a new out of sample data point by training the model on all of the available data and predicting one-step. E.g.:

```
1 yhat = model_fit.forecast()[0]
```



Viral Mehta February 25, 2017 at 2:14 am

Jason, much appreciate your response. I predict same outputs regardless of the observation used in the validation set (last 12 months that the model has seen). This gives me a very nice RMSE every time. If the refit the output for next twelve months regardless of my input effect very quickly if you also change your y variable. Good for one period forward and needs to continue for a period.

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE



Jason Brownlee February 25, 2017

In this specific case we are using walk-forward validation with a model trained for one-step forecasts.

This means as we walk over the validation dataset, observations are moved from validation into training and the model is refit. This is to simulate a new observation being available after a prediction is made for a time step and us being able to update our model in response.

I think what you are talking about is a multi-step forecast, e.g. fitting the model on all of the training data, then forecasting the next n time steps. This is not the experimental setup here, but you are welcome to try it.



Benson Dube March 15, 2017 at 9:11 am #

REPLY ↗

Hello Jason,

If you don't mind me asking, where exactly should the line below fit?:

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

Your Start in Machine Learning



Jason Brownlee March 16, 2017 at 7:57 am #

REPLY ↗

Hi Benson,

Fit the ARIMA model on all of your data.

After that, make a forecast for the next time step by calling the `forecast()` function.



Juanlu February 27, 2017 at 10:15 am #

REPLY ↗

Time to upgrade to matplotlib 2.0, the colors are nicer 😊



Jason Brownlee February 28, 2017 at 8:09 am #

I agree Juanlu! I have recently upgraded to



Hugo March 24, 2017 at 2:38 am #

Dear Jason,

Thanks for the post. Clear, concise and working.

Have you considered to forecast the TS using a SARIM

adding it latter? As a matter of fact, statsmodel has it integrated in its dev version.

(<http://www.statsmodels.org/dev/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html>)

I am wondering if it is possible to take into account hierarchy, like the forecast of monthly sales divided in sub-forecasts of french champagne destination markets, using python.

Thanks and keep posting!

Your Start in Machine Learning

You can master applied Machine Learning
without the math or fancy degree.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Jason Brownlee March 24, 2017 at 7:59 am #

REPLY ↗

Hi Hugo, yes I am familiar with SARIMA, but I decided to keep the example limited to ARIMA.

Yes, as long as you have the data to train/verify the model. I would consider starting with independent models for each case, using all available data and tease out how to optimize it further after that.



Luis Ashurei May 2, 2017 at 7:25 pm #

REPLY ↗

Thanks for the wonderful post Jason,

Your Start in Machine Learning

Two quick question:

1.

When I'm doing Grid Search ARIMA Hyperparameters with your example, it's quite slow on my machine, spent about 1 minutes. However the parameter range and data are not large at all. Is it slow specific on my end? I'm concerning the model is too slow to use.

2.

Does ARIMA support multi-step forecast? E.g. What if I keep use predicted value into next forecast, will it overfitting?

Thanks!



Jason Brownlee May 3, 2017 at 7:34 am #

Your timing sounds fine. If it is too slow for data.

You can do multi-step forecasting directly (forecast) <http://machinelearningmastery.com/multi-step-time>



Luis May 3, 2017 at 6:50 pm #

Thanks in advice.



Pramod Gupta July 3, 2017 at 7:58 pm #

REPLY ↗

Hello Jason

Thanks for this awesome hands-on on Time series. However I have a query.

I extracted year and month from date column as my features for model. Then I built a Linear Regression model and a Random Forest model. My final prediction was a weighted average of both these models. I got an rmse of 366 (similar to yours i.e. 361 on validation data).

Can this approach be the alternative for an ARIMA model? What can be the possible drawbacks of this approach?

Appreciate your comments

Thanks



Jason Brownlee July 6, 2017 at 10:01 am #

REPLY ↗

Nice work. ARIMA may be a simpler mod

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Roberto July 5, 2017 at 7:26 am #

REPLY ↗

Hi just a silly note have you consider using itertools for your evaluate_models() function?

<https://pastebin.com/w0apb6Tg>

in python nested loops are not so readable.



Jason Brownlee July 6, 2017 at 10:21 am #

REPLY ↗

Thanks for the suggestion.



Rishi Patil July 8, 2017 at 9:15 am #

X

Jason, I just finished reading "Introduction to two questions:

- * Are you planning to come out with a "Intermediate co
- * You spend the first few chapters discussing how to re Learning problem using Lag features. However, in late the necessity of using explicitly generated lag features timeseries features with other ML algorithms (such as (I tried out a few AR lags with XGB and got decent results using but coulan t figure out how to incorporate the MA parts).

Would appreciate any insights.

Your Start in Machine Learning

You can master applied Machine Learning without the math or fancy degree.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Jason Brownlee July 9, 2017 at 10:50 am #

REPLY ↗

I may have a more advanced TS book in the future.

Great question. You would have to calculate the MA features manually.



Jan August 2, 2017 at 8:34 pm #

REPLY ↗

Hi Jason,

can it be that TimeGrouper() does not work if there are months missing in a year? Also there are no docs available for TimeGrouper(). May you can use pd.Grouper in your future examples?

Regards

Your Start in Machine Learning



Jason Brownlee August 3, 2017 at 6:49 am #

REPLY ↗

I think it does work with missing data, allowing you to resample.

TimeGrouper does not have docs, but Grouper does and is a good start:

<https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Grouper.html>



Sambit August 17, 2017 at 5:09 am #

REPLY ↗

Hi,

Thanks for this great link on time series.

I am not able to acces the dataset stored at:- <https://data.world/mwchampagne/sales-millions-64-72#!ds=22r5&display=link>

Regards,

Sambit



Jason Brownlee August 17, 2017 at 6:49 am #

I'm sorry to hear that, here is the full dataset:

```

1 "Month", "Sales"
2 "1964-01", 2815
3 "1964-02", 2672
4 "1964-03", 2755
5 "1964-04", 2721
6 "1964-05", 2946
7 "1964-06", 3036
8 "1964-07", 2282
9 "1964-08", 2212
10 "1964-09", 2922
11 "1964-10", 4301
12 "1964-11", 5764
13 "1964-12", 7312
14 "1965-01", 2541
15 "1965-02", 2475
16 "1965-03", 3031
17 "1965-04", 3266
18 "1965-05", 3776
19 "1965-06", 3230
20 "1965-07", 3028
21 "1965-08", 1759
22 "1965-09", 3595
23 "1965-10", 4474
24 "1965-11", 6838
25 "1965-12", 8357
26 "1966-01", 3113
27 "1966-02", 3006
28 "1966-03", 4047
29 "1966-04", 3523
30 "1966-05", 3937

```

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Your Start in Machine Learning

```

31 "1966-06",3986
32 "1966-07",3260
33 "1966-08",1573
34 "1966-09",3528
35 "1966-10",5211
36 "1966-11",7614
37 "1966-12",9254
38 "1967-01",5375
39 "1967-02",3088
40 "1967-03",3718
41 "1967-04",4514
42 "1967-05",4520
43 "1967-06",4539
44 "1967-07",3663
45 "1967-08",1643
46 "1967-09",4739
47 "1967-10",5428
48 "1967-11",8314
49 "1967-12",10651
50 "1968-01",3633
51 "1968-02",4292
52 "1968-03",4154
53 "1968-04",4121
54 "1968-05",4647
55 "1968-06",4753
56 "1968-07",3965
57 "1968-08",1723
58 "1968-09",5048
59 "1968-10",6922
60 "1968-11",9858
61 "1968-12",11331
62 "1969-01",4016
63 "1969-02",3957
64 "1969-03",4510
65 "1969-04",4276
66 "1969-05",4968
67 "1969-06",4677
68 "1969-07",3523
69 "1969-08",1821
70 "1969-09",5222
71 "1969-10",6872
72 "1969-11",10803
73 "1969-12",13916
74 "1970-01",2639
75 "1970-02",2899
76 "1970-03",3370
77 "1970-04",3740
78 "1970-05",2927
79 "1970-06",3986
80 "1970-07",4217
81 "1970-08",1738
82 "1970-09",5221
83 "1970-10",6424
84 "1970-11",9842
85 "1970-12",13076
86 "1971-01",3934
87 "1971-02",3162
88 "1971-03",4286
89 "1971-04",4676
90 "1971-05",5010
91 "1971-06",4874
92 "1971-07",4633
93 "1971-08",1659
94 "1971-09",5951
95 "1971-10",6981

```

Your Start in Machine Learning

You can master applied Machine Learning without the math or fancy degree.

Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)

Your Start in Machine Learning

```

96 "1971-11",9851
97 "1971-12",12670
98 "1972-01",4348
99 "1972-02",3564
100 "1972-03",4577
101 "1972-04",4788
102 "1972-05",4618
103 "1972-06",5312
104 "1972-07",4298
105 "1972-08",1413
106 "1972-09",5877

```



sarrauste August 24, 2017 at 1:15 am #

REPLY ↗

hi,

i am having the following error:

X = X.astype('float32')

ValueError: could not convert string to float: '1972-09'

can you please help me

ty



Jason Brownlee August 24, 2017 at 6:44 am

Looks like you may have missed some lin

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Nico October 29, 2018 at 5:02 am #

REPLY ↗

Hi Jason, I am having the same error. I checked the lines and I have them all. How can python convert 'yyyy-mm' to float32?

I dowload the csv from the link of datamarket you gave above.

What am i missing?

Thnanx



Jason Brownlee October 29, 2018 at 6:04 am #

REPLY ↗

I have some suggestions here:

<https://machinelearningmastery.com/faq/single-faq/why-does-the-code-in-the-tutorial-not-work-for-me>

Your Start in Machine Learning



sandip August 28, 2017 at 8:09 pm #

REPLY ↗

Hi Jason , Nice approach for time series forecasting.Thanks for this Article.

Just had a one small doubt,Here we are tuning our ARIMA parameter on test data using RMSE as a measure but generally we tune our parameter on the training(train+validation) data and then we check whether those parameters are well generalized or not by using test data. is it right?

Because my concern is that if we choose our parameter on that particular test data,whether those parameter will generalized to coming new test data or not? I feel that there will be biasness while choosing parameter because we are specifically choosing parameter those giving less RMSE for that test data. Here we are not checking whether our model is working/fitted well for our train data or not?

If i am wrong just correct me.Let me know the logic behind this parameter tuning on the basis of test data?



Jason Brownlee August 29, 2017 at 5:03 pm

Yes, that is a less biased way to prepare a model.

More here:

<https://machinelearningmastery.com/difference-test/>



sandip August 29, 2017 at 5:58 pm #

Thank you so much Jason.



Jason Brownlee August 30, 2017 at 6:13 am #

REPLY ↗

You're welcome sandip.



Anthy August 31, 2017 at 2:24 am #

REPLY ↗

Hi Jason,

Thank you for this detailed and clear tutorial.

I have a question concerning finding ARIMA's parameters. I didn't understand why we have an imbricated "for" loop and we do consider all the p values and not all the q values.

Please ask if the question is not that clear.

Thank you in advance.

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Jason Brownlee August 31, 2017 at 6:23 am #

REPLY ↗

Please review the example again, we grid search p, d and q values.



Ella Zhao September 2, 2017 at 4:26 am #

REPLY ↗

Hi Jason,

Thanks for this awesome example! It helped me a lot! However if you don't mind, I have a question on the prediction loop.

Why did you use history.append(obs) instead of history.append(yhat)? It seems like you add the observation (real data in test) to yhat, months_in_year is based on the history data. but solving practical problem.

I've tried to use history.append(yhat) in my model, but

Appreciate your comments

Thanks,

Ella

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Jason Brownlee September 2, 2017 at 6:16 am #

Good question, because in this scenario, we are assuming that each new real observation is available after we make a prediction.

Pretend we are running this loop one iteration per day, and we get yesterdays observation this morning so we can accurately predict tomorrow.



Ella Zhao September 2, 2017 at 11:46 pm #

REPLY ↗

Thank you so much for your explanation!!



Jason Brownlee September 3, 2017 at 5:45 am #

REPLY ↗

You're welcome.

Your Start in Machine Learning



Alaoui September 3, 2017 at 12:27 am #

REPLY ↗

Hello,

Thanks a lot for this work.

I have a point to discuss on concerning the walk forward validation.

I saw that you fill the history part by a value from the test set at each iteration :

....

```
# observation
```

```
obs = test[i]
```

```
history.append(obs)
```

....

Do we really have to do this or we have to use the new
Indeed when we have to do a future prediction on some
Regards and thanks in advance for your feedback

Simo



Jason Brownlee September 3, 2017 at 5:48 am #

X

Your Start in Machine Learning

You can master applied Machine Learning
without the math or fancy degree.

Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

It depends on your model and project goals.
In this case, we are assuming that the true observations will be used in the formulation of the next prediction, a

If you use outputs as inputs (e.g. recursive multi-step forecasting), error will compound and things will go more crazy, sooner (e.g. model skill will drop). See this post:

<https://machinelearningmastery.com/multi-step-time-series-forecasting/>



Alaoui September 3, 2017 at 12:49 am #

REPLY ↗

Sorry, I just read the Ella question...So for long range future prediction we have to use the yhat...

Thanks



Jason Brownlee September 3, 2017 at 5:48 am #

REPLY ↗

You can if you choose.



Sujeeet September 16, 2017 at 12:22 pm #

REPLY ↗

Your Start in Machine Learning

Hi,

I am getting NAN after these steps:

```
from pandas import Series
series = Series.from_csv('champagne.csv', header=0)
print(len(series))
split_point = len(series) - 12
print(split_point)
dataset, validation = series[0:split_point], series[split_point:]
print(dataset)
print('Dataset %d, Validation %d' % (len(dataset), len(validation)))
dataset.to_csv('dataset.csv')
validation.to_csv('validation.csv')
```



Jason Brownlee September 17, 2017 at 5:22

I'm sorry to hear that.

Ensure that you have copied all of the code and the
Also confirm that your environment is up to date.



Jaryn September 20, 2017 at 3:24 am #

Hi Jason,

This is extremely useful. Thank you very much!

I was wondering if you could provide the full code to extend the forecast for 1 year ahead. I know you mentioned above that there is a forecast function but when I run:

```
yhat = model_fit.forecast()[0]
pyplot.plot(y)
pyplot.plot(yhat, color='green')
pyplot.plot(predictions, color='red')
pyplot.show()
```

I don't get any green lines in my code. I'm sure I'm missing a lot of things here, but I don't know what.

Thank you again, much appreciated!



Jason Brownlee September 20, 2017 at 6:01 am #

REPLY ↗

This post will help you with making longer-term forecasts:

<https://machinelearningmastery.com/make-sample/>

Your Start in Machine Learning



Kunal Shrivastava September 20, 2017 at 4:54 pm #

REPLY ↗

Hi Jason,

I am using this modelling steps to model my problem. When I am passing best parameter which i have got from Grid Search for doing prediction, I am getting Linear Algebra error and sometimes not converging Due to this I am not able to predict the values.

My second concern is if this is happening for Grid search, how can i automate the script in production environment. Lets say i have to forecast some values for 50000 different sites then what is best way to achieve this goal ?



Jason Brownlee September 21, 2017 at 5:36 pm #

Sorry to hear that, perhaps you have not specified the parameters for the grid search?

I have some notes about models in production here:
<http://machinelearningmastery.com/deploy-machine-learning/>



Tri Bien September 24, 2017 at 7:57 pm #

Hi Jason,

Thank for your post, it is really helpful for me. It is great!

I have a small question:

In the post, you use 2 data set: dataset.csv and validation.csv

+ In dataset.csv, you split it and use Walk-forward validation -> I totally agree.

+ In validation.csv, you still re-train ARIMA model with walk-forward validation (model_fit = model.fit(trend='nc', disp=0) – line 45 in section 7.3) -> I think it is unseen data for testing, so we don't train model here and only test the model's performance. Is it correct?

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Jason Brownlee September 25, 2017 at 5:38 am #

REPLY ↗

In the case of walk forward validation, we are assuming that the real observations are available after the time of prediction.

You can make different assumptions when evaluating your model.



Navid September 30, 2017 at 1:50 am #

REPLY ↗

Your Start in Machine Learning

Hi, Jason

First of all, thank you for your wonderful tutorial.

Until now (part 5.4) I just have one doubt: why you plotted a histogram/kde without removing the trend first? If the goal is to see how the distribution shape is similar to Gaussian distribution, doesn't a trend changes the distribution of data?

Thanks

Navid



Jason Brownlee September 30, 2017 at 7:44 am #

REPLY ↗

Great question, at that point we are early density plots before and after making the series stationary.

Trend removal generally won't impact the form of the distribution, just the scale of operations.



Nick October 19, 2017 at 12:46 am #

Hi Jason,
Thanks for your awesome post. Could you explain how to remove the trend from the data? I have 1-year data? My dataset is recorded in half an hour intervals. The sales are large in summer and small in winter, i.e. in winter it's between 5000-14000.

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Shashank Hegde October 22, 2017 at 3:46 pm #

REPLY ↗

Hi Jason,

Thank you for the post.I want to implement 'ARIMA' function instead of using built-in function.
Do you know where I can find the algorithm to implement 'ARIMA' function as well understanding that in detail manner?



Jason Brownlee October 23, 2017 at 5:41 am #

REPLY ↗

A good textbook on the math is this one: <http://amzn.to/2zvfIcB>



Hara October 28, 2017 at 3:38 am #

REPLY ↗

Hi Jason,

Your Start in Machine Learning

Thanks for the brief tutorial on Time Series forecasting. I am receiving the error “Given a pandas object and the index does not contain dates” when running the ARIMA model code snippet.



Jason Brownlee October 28, 2017 at 5:16 am #

REPLY ↗

Be sure you copied all of the sample code exactly including indenting, and also be sure you have prepared the data, including removing the footer information from the file.



Makis December 18, 2017 at 8:27 pm #

REPLY ↗

Dear Jason,

Thanks for your amazing tut!

I have one problem only when i run the rolling forecast

I use a different dataset which has temperature values first four days and i use the last day as a test dataset. I observation to the history and my arima returns the foll AR coefficients are not ”

ValueError: The computed initial AR coefficients are not You should induce stationarity, choose a different model pass your own start_params.

for i in range(0, len(x_test)):

```
# difference data
diff = difference(history, minutes)
# predict
model = importer.ARIMA(diff, order=(5, 0, 1))
model_fit = model.fit(trend='nc', disp=0)
yhat = model_fit.forecast()[0]
yhat = inverse_difference(history, yhat, minutes)
predictions.append(yhat)
# observation
obs = x_test[i]
history.append(obs)
print('Predicted=%f, Expected=%f' % (yhat, obs))
```

My x_test dataset contains 1440 rows and i get the error on the 1423 iteration of the loop. Until iteration 1423 the each arima model does not have issues.

Your help is precious to me.

Thanks again!

Your Start in Machine Learning

You can master applied Machine Learning without the math or fancy degree.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Kindest Regards,
Makis



Jason Brownlee December 19, 2017 at 5:17 am #

REPLY ↗

Have you confirmed that your data set is stationary? Perhaps there is still some trend or seasonal structure?

I have many posts on detrending and seasonal adjustment that may help.



Makis December 20, 2017 at 12:54 am #

Dear Jason,

Thanks for your response!

I have used a different hyperparameter (Arima 0,1,1) and this is the right thing, since 5,0,1 and 5,1,1 had that? And please, one final question, my result is 0,01. The prediction line in the plot is almost overlapping history.append(obs)? I'm not sure i understand what history is. And what is the difference of doing a prediction compared by using the steps parameter in 1 month?

Sorry for the long questions!

Your tutorials are even better than the books i'm currently reading!

Cheers from Greece.

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Jason Brownlee December 20, 2017 at 5:47 am #

REPLY ↗

Well done. I'd recommend using the model that is skillful and stable.



Makis December 20, 2017 at 8:22 am #

Hey Jason! What i'm trying to say is, maybe the good predictions are because i append the y observation to the history. Why we do this? what is the purpose? If we do not append the y observation then the results are going to be still valid?

Your Start in Machine Learning



Jason Brownlee December 20, 2017 at 3:49 pm #

In the test setup we are assuming that real observations are made after each prediction that we can in turn use to make the next prediction.

Your specific framing of the problem may differ and I would encourage you to design a test harness to capture that.



Makis December 20, 2017 at 10:42 pm #

Hey Jason, for once more than I will follow your suggestion right now

Thanks for everything

Kind Regards,
Makis



Makis December 20, 2017 at 10:51 pm #

Hey Jason, one last question

Since i do the rolling forecast manner
do i need to save a model and do a the

Can i simply do the rolling forecast loop to start from 0 instead of 1?

Your Start in Machine Learning

You can master applied Machine Learning without the math or fancy degree.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Jason Brownlee December 21, 2017 at 5:26 am #

Sure.



Amir December 19, 2017 at 6:51 pm #

REPLY ↗

Hi Jason,

Thank you so much for this amazing tutorial. It really helps me to learn time series forecasting and prepare for my new job. I'm a total beginner in data analysis/science as I'm currently making transition from engineering to data analysis.

But the problem that I'm facing with this tutorial is as I'm running the code, I'm stuck with the final step, "Validate the Model". I've tried to re-copy and re-run the sample code many times but it didn't seem to work. Here is the error.

Your Start in Machine Learning

```
>Predicted=10101.763, Expected=9851
>Predicted=13219.067, Expected=12670
>Predicted=3996.535, Expected=4348
>Predicted=3465.934, Expected=3564
>Predicted=4522.683, Expected=4577
>Predicted=4901.336, Expected=4788
>Predicted=5190.094, Expected=4618
>Predicted=4930.190, Expected=5312
>Predicted=4944.785, Expected=4298
>Predicted=1699.409, Expected=1413
>Predicted=6085.324, Expected=5877
>Predicted=7135.720, Expected=nan
```

ValueError Traceback (most recent call last)

in ()

56

57 # Report performance

—> 58 mse = mean_squared_error(y, predictions)

59 rmse = sqrt(mse)

60 print('RMSE: %.3f' % rmse)

```
/Users/amir/Library/Enthought/Canopy edm/envs/User
packages/sklearn/metrics/regression.py in mean_squa
229 """
230 y_type, y_true, y_pred, multioutput = _check_reg_
231 y_true, y_pred, multioutput)
```

```
232 output_errors = np.average((y_true - y_pred) ** 2,
233 weights=sample_weight)
```

```
/Users/amir/Library/Enthought/Canopy edm/envs/User/lib/python3.5/site-
packages/sklearn/metrics/regression.py in _check_reg_targets(y_true, y_pred, multioutput)
73 """
74 check_consistent_length(y_true, y_pred)
```

—> 75 y_true = check_array(y_true, ensure_2d=False)

76 y_pred = check_array(y_pred, ensure_2d=False)

77

```
/Users/amir/Library/Enthought/Canopy edm/envs/User/lib/python3.5/site-packages/sklearn/utils/validation.py
in check_array(array, accept_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd,
ensure_min_samples, ensure_min_features, warn_on_dtype, estimator)
405 % (array.ndim, estimator_name))
```

406 if force_all_finite:

—> 407 _assert_all_finite(array)

408

409 shape_repr = _shape_repr(array.shape)

```
/Users/amir/Library/Enthought/Canopy edm/envs/User/lib/python3.5/site-packages/sklearn/utils/validation.py
in _assert_all_finite(X)
56 and not np.isfinite(X).all()):
```

Your Start in Machine Learning

You can master applied Machine Learning
without the math or fancy degree.

Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)

```

57 raise ValueError("Input contains NaN, infinity"
→ 58 " or a value too large for %r." % X.dtype)
59
60

```

ValueError: Input contains NaN, infinity or a value too large for dtype('float32').

Could you please help me with this?

Thank you so much

Amir



Jason Brownlee December 20, 2017 at 5:41 pm #

I'm not sure of the cause of your fault, so

Ensure your libraries are up to date and that you have



Ankit Tripathi April 30, 2018 at 7:24 pm #

The csv file contains some jargon text which



Satyajit Pattnaik December 28, 2017 at 5:03 pm #



Your Start in Machine Learning

You can master applied Machine Learning
without the math or fancy degree.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

REPLY

Hi Jason,

In the prediction area, where you have added the observation to history and then running the loop to find the ARIMA results.

i.e.

```

model = ARIMA(history, order=(4,1,2))
model_fit = model.fit(disp=0)
output = model_fit.forecast()
yhat = output[0]
predictions.append(yhat)
obs = test[t]
history.append(obs)

```

Here, what if we append yhat to history, when i am appending yhat to history, my results are really bad, pls help..

Because as per my model, we need to predict the test data by using only the training data, so we cannot use obs to be appended in history, hope you get my point.

Your Start in Machine Learning



Jason Brownlee December 29, 2017 at 5:18 am #

REPLY ↗

Yes, that would be called a recursive multi-step forecast. It is challenging. You can learn a little more about it here:

<https://machinelearningmastery.com/multi-step-time-series-forecasting/>



Satyajit Pattnaik December 29, 2017 at 4:06 pm #

REPLY ↗

Concept wise i understand what is Recursive Multi step forecasting, that's the same i have used in my earlier reply code, i have appended obs in history. so that means everytime my loop runs, it takes the existing training data + next c correctly, but my results are really bad..

Or do you mean to say, for this we need to plot pdq values, and then run the ARIMA function in a way to determine pdq values in loop



Jason Brownlee December 30, 2017 at 10:30 am #

Generally using forecasts in a recursive manner is creative with your model/models.

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Udi January 8, 2018 at 8:57 pm #

REPLY ↗

Hello Jason,

Thank you for your clear and straightforward post!

I have the same question as Nick above – about the choice of differentiation interval.

In your example you set it to 12 according to the expected cycle of the data.

However, in a more problematic case the data does not seem to imply a clear cycle (ACF and PACF graphs notwithstanding).

In my case, I've found out that setting the interval to 12 yielded better results than my default, which was 1. I can understand why choosing a small interval would be generally bad – random noise is too dominant. I have more difficulty understanding how to calibrate the ideal interval value for my data (except brute force, that is. Maybe I shouldn't calibrate? That could induce overfit. Anyway, I still should find a generally decent value).



Jason Brownlee January 9, 2018 at 5:29 am #

REPLY ↗

Your Start in Machine Learning

If the amount of data is relatively small, consider using a grid search across difference values to see what works.



Udi January 8, 2018 at 9:09 pm #

REPLY ↗

And another question. I applied a grid search in order to choose the hyperparameters of my ARIMA model. I fear that bias correction could be counter productive in this case.

Do you have any insights on this subject? Would you perform both (grid search followed by bias correction)? Is the answer data dependent?



Jason Brownlee January 9, 2018 at 5:29 am

I would only perform bias correction if it lift



Nirmal Kanti Sahoo January 29, 2018 at 11:33 am

Hi Jason,

I have around more than 1000 of products. I have sales data for each product. I want to develop a model by which I can predict sales. Could you please help me how do I proceed with evaluation?

Thanking in advance.

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Jason Brownlee January 30, 2018 at 9:45 am #

REPLY ↗

Perhaps start here:

<https://machinelearningmastery.com/start-here/#timeseries>



Udi February 18, 2018 at 9:32 pm #

REPLY ↗

(Noob's parallelization question, probably)

Does anybody know whether statmodels ARIMA uses multi-threading or any other kind of parallelization?

I'm trying to run an analysis based on multiple ARIMA fits on my laptop and thread parallelization increases total runtime rather than decrease it. It seems a single ARIMA fit (part of a single thread) uses several processors at once....

Your Start in Machine Learning

Thanks,
Udi



Jason Brownlee February 19, 2018 at 9:05 am #

REPLY ↗

I think it is single threaded.



Raul February 28, 2018 at 3:49 am #

REPLY ↗

Hi Jason,

This is an excellent article, and it is super helpful. I had asked but after reading through the comments, I'm not most similar to Nirmals.

1. I have a dataset with multiple "wines", each with the
2. This dataset has other variables that aren't just time

For example,

	Month 1 Sales	Month 2 Sales	Month 3 Sales	online Reviews
Wine A				
Wine B				
Wine C				
Wine D				

I'm wondering if there's an extension of this Time Series model that can take into account other variables and other instances of historical data. Let me know if I should clarify.

Thank you for taking the time to clarify.

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Jason Brownlee February 28, 2018 at 6:10 am #

REPLY ↗

Yes, perhaps you could model each series separately, or groups of series or all together, or all 3 approaches and ensemble the results.

Also, I'd recommend moving on to other models, such as ML or even MLP models.

I hope to cover this topic in more detail soon.



Raul February 28, 2018 at 7:01 am #

REPLY ↗

Thanks for the quick reply.

Your Start in Machine Learning

I'll try out your first method.

But do you have any good ML/MLP models/tutorials to start with? It's okay if you don't! I noticed you have a nice article on multivariate time series here:

<https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/>

I haven't read through it yet, but I think it only takes into account historical data of one instance of multiple variables.

I think it would be interesting to find out how to ensemble the results of my

#1. one instance of historical data (Wine A's last 3 month sales)

#2. one instance of non-historical variables (Wine A's online reviews, type, etc.)

To me, # 1's output is a "variable" to be used in the ML model for #2. If that makes sense, do you think that's the right way of going about things?

Thanks for being so responsive!



Jason Brownlee March 1, 2018 at 10:30 pm #

I hope to cover this topic in more detail in the future, perhaps using a different dataset from many different directions



Anchal April 6, 2018 at 4:20 am #

Thank you Jason! for really making ML practical and easy to understand. I am doing awesome in my ML course.

In this blog you have mentioned that the results suggest that what little autocorrelation is present in the time series has been captured by the model.

What will be the next steps if there is high autocorrelation?

Thanks in advance.



Jason Brownlee April 6, 2018 at 6:35 am #

REPLY ↗

Great question!

Some ideas:

- Perhaps try alternate models.
- Perhaps improve the data preparation.
- Perhaps try downstream models to further correct the forecast.

Ankit Tripathi May 10, 2018 at 5:55 pm #

Your Start in Machine Learning



Hey Jason,

That was a very well informed article! I am trying to forecast on a weekly data. Any tips for improving the model since weekly data is hard to forecast. Also should “months_in_year = 12” used in differencing be changed to “weeks in month=4” to accommodate for weekly frequency?

Thanks



Jason Brownlee May 11, 2018 at 6:34 am #

REPLY ↗

My best tips are to try lots of data preparation techniques and tune the ARIMA using a grid search.



Maria Dossin May 26, 2018 at 9:47 am #

Hi Jason,
Thank you for an amazing tutorial!
Just one quick question: can you please provide solutions?
– Load the model and use it to forecast the next 12 months
will quickly start to degrade in skill.
Thank you~

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Jason Brownlee May 27, 2018 at 6:43 am #

REPLY ↗

Yes, this will help:

<https://machinelearningmastery.com/make-sample-forecasts-arima-python/>



Piyasi Choudhury June 2, 2018 at 11:38 am #

REPLY ↗

Hi Jason

My time series data is non stationary as well and I tried upto 3rd degree of differentiation after which I cant proceed any more as the data is exhausted. It has 3 calculated points and on doing Fuller test for checking stationarity, it errors out “maxlag should be < nobs”. What can I do here?



Jason Brownlee June 3, 2018 at 6:20 am #

REPLY ↗

Sounds like you might be running out of data in your series. Perhaps you need more observations?

Your Start in Machine Learning



KACEM June 3, 2018 at 11:54 am #

REPLY ↗

Hello, thank you.

i wonder if we can say that python give good forecasts than other forecast logiciels like Aperia.

To forecast sales we have to base ourselves in historic and baseline but in fact, in industry there are many factors that complexify the calculation of our forecast like the marketing actions, exceptional customer order, ... How can we consider these perturbation in our forecasts to ensure good value of accuracy.

Thank you for answering.



Jason Brownlee June 4, 2018 at 6:21 am #

What is Aperia?

Additional elements could be used as exogenous variables.



cathy June 23, 2018 at 10:00 am #

Jason,

The dataset I have requires to double difference in order to be stationary.

```
diff1 = difference(history, 12)
diff2 = difference(diff1, 12),
```

it worked and made it stationary with ADF test, however, how do I reverse it back please?

Thank you

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Jason Brownlee June 24, 2018 at 7:27 am #

REPLY ↗

Add the values back. It requires that you keep the original and the first diffed data.



Cathy June 24, 2018 at 10:28 am #

REPLY ↗

Thanks, so would the function look like this?

```
yhat = inverse_difference(diff1, yhat, interval)
yhat = inverse_difference(history, yhat, interval) ?
```

Thank you!

Your Start in Machine Learning



Jason Brownlee June 25, 2018 at 6:16 am #

REPLY ↗

Correct.



Rui July 27, 2018 at 4:11 am #

REPLY ↗

First of all, this is a great article, well written and detailed. I have a question regarding on the use of the test set on all of the analysis. What about putting this model in production? You wouldn't have the 'test[i]' (or 'y[i]') for each iteration to add to the list 'history' to have it as part of the training set? My point is that that instead of adding the values from the test set to the training set, you would have predictions being made to the training set in order to do a final prediction. Thanks again for the resource and all the help putting this together.



Jason Brownlee July 27, 2018 at 5:57 am #

A final model will be created by training it on the training set and then making predictions in the future.

You can learn more about final models here:

<https://machinelearningmastery.com/train-final-machine-learning-model/>

You can learn more about making out of sample predictions here:

<https://machinelearningmastery.com/make-sample-forecasts-arima-python/>



James Lucas July 27, 2018 at 2:21 pm #

REPLY ↗

I keep getting errors

```
TypeError Traceback (most recent call last)
in ()
9 obs = test[i]
10 history.append(obs)
→ 11 print('Predicted=%3f, Expected=%3.f' % (yhat, obs))
```

TypeError: must be real number, not ellipsis

IE: # predict

yhat = ...

those 3 dots (ellipsis) are throwing errors.

The code has ... throughout. Any idea how to fix this error?

Your Start in Machine Learning

You can master applied Machine Learning without the math or fancy degree.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Jason Brownlee July 28, 2018 at 6:28 am #

REPLY ↗

The code with “...” is just example code, it is not for execution. Skip it.

Perhaps a more careful read of the tutorial is in order James?



Ben Kesby August 3, 2018 at 2:53 am #

REPLY ↗

I'm having an issue with the PACF plot for the residuals in part 6.3 Plotting The Residuals. I have followed your code exactly with the ACF plot resulting in values near 1. However, the corresponding PACF is plotting values nearing 8 at around 36 – 38 lags.



JP August 24, 2018 at 8:17 pm #

Hi Jason,
Thanks so much for this tutorial. One thing though. As you can see from my screenshot, I am lost when opening the dataset, for example when trying to convert the date column. I am looking for a workaround that would keep the date format while still being able to use the .groupby() function.
Thanks!

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Jason Brownlee August 25, 2018 at 5:48 am #

REPLY ↗

Yes you can use `pandas.read_csv()` with the same arguments.



Alexandra September 12, 2018 at 8:40 pm #

REPLY ↗

How can I improve the readability of Seasonal Per Year Line Plots (especially when it comes to axes)? I would be grateful for your help.



Jason Brownlee September 13, 2018 at 8:01 am #

REPLY ↗

Perhaps create one plot per figure?



Alexandra September 13, 2018 at 4:44 pm #

Your Start in Machine Learning

I'd rather have a comparison between all the subplots.



Jason Brownlee September 14, 2018 at 6:33 am #

REPLY ↗

Perhaps plot them all on the same figure?

Perhaps calculate the different and plot that?



Chintan September 18, 2018 at 2:16 am #

REPLY ↗

Hi Jason,

I was able to predict and chart looks nice. Just wonder
I wanted to see for next month prediction how are we a

Thanks,
Chintan



Jason Brownlee September 18, 2018 at 6:20 pm #

Your Start in Machine Learning

You can master applied Machine Learning
without the math or fancy degree.

Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE



Pranay October 9, 2018 at 9:26 pm #

REPLY ↗

Hi Jason, I deal with daily set data. What is the issue when I try months_in_year = 364?? I mean it's not throwing best_arima out when I set months_in_year = 364. May I know the reason?



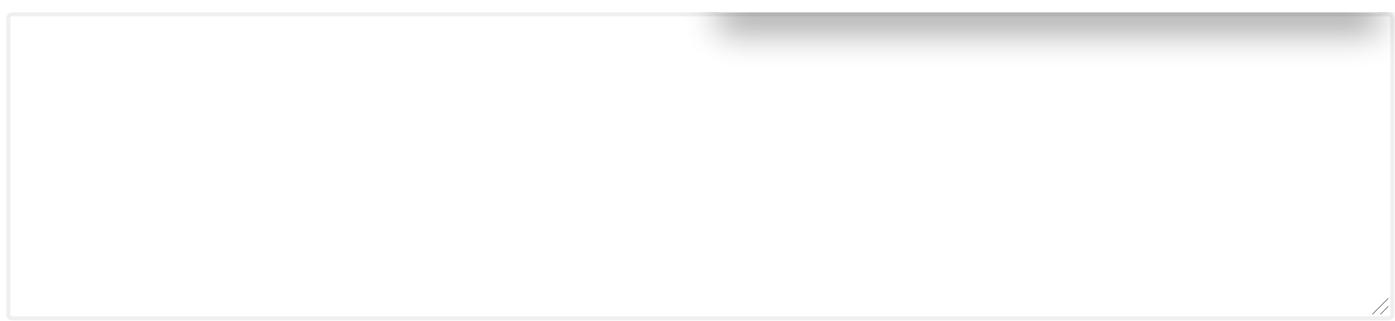
Jason Brownlee October 10, 2018 at 6:09 am #

REPLY ↗

Why are you making that change? I don't follow?

Leave a Reply

Your Start in Machine Learning



Name (required)

Email (will not be published) (requi

Website

SUBMIT COMMENT

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

Email Address

[START MY EMAIL COURSE](#)



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.

[Your Start in Machine Learning](#)

[Click to Get Started Now!](#)

Introduction to Time Series Forecasting with Python

How to Prepare Data and Develop Models to Predict the Future

Jason Brownlee

MACHINE
LEARN
MAS

Your Start in Machine Learning

X

You can master applied Machine Learning **without the math or fancy degree.**

Find out how in this *free* and *practical* email course.

[START MY EMAIL COURSE](#)

POPULAR



[How to Develop a Deep Learning Photo Caption Generator](#)

NOVEMBER 27, 2017



[How to Develop a Neural Machine Translation System](#)

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 Start in 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

© 2018 Machine Learning Mastery. All Rights Reserved.

[Privacy](#) | [Disclaimer](#) | [Terms](#) | [Contact](#)

Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**. Find out how in this *free* and *practical* email course.

START MY EMAIL COURSE

Your Start in Machine Learning