Advanced Forecasting with Python

With State-of-the-Art-Models Including LSTMs, Facebook's Prophet, and Amazon's DeepAR

—

Joos Korstanje

Apress®

Advanced Forecasting with Python

With State-of-the-Art-Models Including LSTMs, Facebook's Prophet, and Amazon's DeepAR

Joos Korstanje

Apress®

Advanced Forecasting with Python: With State-of-the-Art-Models Including LSTMs, Facebook's Prophet, and Amazon's DeepAR

Joos Korstanje
Maisons Alfort, France

ISBN-13 (pbk): 978-1-4842-7149-0 ISBN-13 (electronic): 978-1-4842-7150-6
https://doi.org/10.1007/978-1-4842-7150-6

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Celestin Suresh John
Development Editor: Matthew Moodie
Coordinating Editor: Aditee Mirashi

Cover designed by eStudioCalamar

Cover image designed by Freepik (www.freepik.com)

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, Suite 4600, New York, NY 10004-1562, USA. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-7149-0. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

This book is dedicated to my partner, Olivia,
for the help and support throughout the period of writing.

Table of Contents

About the Author

Joos Korstanje is a data scientist, with over five years of industry experience in developing machine learning tools, of which a large part is forecasting models. He currently works at Disneyland Paris where he develops machine learning for a variety of tools. His experience in writing and teaching has motivated him to write this book, *Advanced Forecasting with Python*.

About the Technical Reviewer

Michael Keith is a data scientist working in the public health sector based in Salt Lake City, Utah. He is passionate about using data to improve health and educational outcomes and is a lead forecaster for the Utah Department of Health, leveraging Python to produce hundreds of forecasts every month. He earned a master's degree from Florida State University and has worked in data-related roles for several organizations, including Disney in Orlando. He has produced data science–themed videos for Apress, writes for *Towards Data Science*, performs consultations for Western Governors University, and lectures annually to graduate students at Florida State. In his free time, he enjoys road biking, hiking, and watching movies with his wife and beautiful 7-month-old daughter.

Introduction

Advanced Forecasting with Python covers all machine learning techniques relevant for forecasting problems, ranging from univariate and multivariate time series to supervised learning, to state-of-the-art deep forecasting models like LSTMs, Recurrent Neural Networks (RNNs), Facebook's open source Prophet model, and Amazon's DeepAR model.

Rather than focus on a specific set of models, this book presents an exhaustive overview of all techniques relevant to practitioners of forecasting. It begins by explaining the different categories of models that are relevant for forecasting in a high-level language. Next, it covers univariate and multivariate time series models followed by advanced machine learning and deep learning models, such as Recurrent Neural Networks, LSTMs, Facebook's Prophet, and Amazon's DeepAR. It concludes with reflections on model selection like benchmark scores vs. understandability of models vs. compute time and automated retraining and updating of models. Each of the models presented in this book is covered in depth, with an intuitive simple explanation of the model, a mathematical transcription of this idea, and Python code that applies the model to an example dataset.

This book is a great resource for those who want to add a competitive edge to their current forecasting skillset. The book is also adapted to those who start working on forecasting tasks and are looking for an exhaustive book that allows them to start with traditional models and gradually move into more and more advanced models.

You can follow along with the code using the GitHub repository that contains a Jupyter notebook per chapter. You are encouraged to use Jupyter notebooks for following along, but you can also run the code in any other Python environment of your choice.

PART I

Machine Learning for Forecasting

CHAPTER 1

Models for Forecasting

Forecasting, grossly translated as the task of predicting the future, has been present in human society for ages. Whether it is through fortune-tellers, weather forecasts, or algorithmic stock trading, man has always been interested in predicting what the future holds.

Yet forecasting the future is not easy. Consider fortune-tellers, stock market gurus, or weather forecasters: many try to predict the future, but few succeed. And for those who succeed, you will never know whether it was luck or skill.

In recent years, the computing power of computers has become much more commonly available than, say, 30 years ago. This has created a great boom in the use of Artificial Intelligence. Artificial Intelligence and especially machine learning can be used for a wide range of tasks, including robotics, self-driving cars, but also forecasting, that is, if you have a reasonable amount of data about the past that you can project into the future.

Throughout this book, you will learn the modern machine learning techniques that are relevant for forecasting. I will present a large number of machine learning models, together with an intuitive explanation of the model, its mathematics, and an applied use case.

The goal of this book is to give you a real insight into the application of those machine learning models. You will see worked examples applied to real datasets together with honest evaluations of the results: some successful, some less successful.

In this way, this book is different than many other resources, which often present perfectly fitting use cases on simulated data. To learn real-life machine learning and forecasting, it is important to know how models work, but it is even more important to know how to evaluate a model honestly and objectively. This pragmatical point of view will be the guideline throughout the chapters.

J. Korstanje, *Advanced Forecasting with Python*, https://doi.org/10.1007/978-1-4842-7150-6_1

Reading Guide for This Book

Before going further into the different models throughout this book, I will first present a general overview of the machine learning landscape: many types and families of models exist. Each of them has its applications. Before starting, it is important to have an overview of the types of models that exist in machine learning and which of them are relevant for forecasting.

After this, I will cover several strategies and metrics for evaluating forecasting models. It is important to understand objective evaluation before practicing: you need to understand your goal before starting to practice.

The remaining chapters of the book will each cover a specific model with an intuitive explanation of the model, its mathematical definitions, and an application on a real dataset. You will start simple with common but simple methods and work your way up to the most recent and state-of-the-art methods on the market.

Machine Learning Landscape

Having the bigger picture of machine learning models before getting into detail will help you to understand how the different models compare to each other and will help you to keep the big picture throughout the book. You will first see univariate time series and supervised regression models: the main categories of forecasting models. After that, you will see a shorter description of machine learning techniques that are less relevant for forecasting.

Univariate Time Series Models

The first category of machine learning models that I want to talk about is time series models. Even though univariate time series have been around for a long time, they are still used. They also form an important basis for several state-of-the-art techniques. They are classical techniques that any forecaster should be familiar with.

Time series models are models that make a forecast of a variable by looking only at historical developments of the variable itself. This means that time series, as opposed to other model families, do not try to describe any "logical" relationships between variables. They do not try to explain the "why" of trends or seasonalities, but they simply put a mathematical formula on the past and try to project it to the future.

Time series modeling is sometimes criticized for this "lack of science." But time series models have gained an important place in forecasting due to their performances, and they could not be ignored.

A Quick Example of the Time Series Approach

Let's look at a super-simple, purely hypothetical example of forecasting the average price of a cup of coffee in an imaginary city called X. Imagine someone has made the effort of collecting the average price of coffee for 90 years in this town, with intervals of five years, and that this has yielded the data in Table 1-1.

Table 1-1. *A Hypothetical Example: The Price of a Cup of Coffee Over the Years*

Year	Average Price
1960	0.80
1965	1.00
1970	1.20
1975	1.40
1980	1.60
1985	1.80
1990	2.00
1995	2.20
2000	2.40
2005	2.60
2010	2.80
2015	3.00
2020	3.20

This fictitious data clearly shows an increase of 20 cents in the price every five years. This is a **linear increasing trend**: linear because it increases with the same amount every year and increasing because it becomes more rather than less.

Let's get this data into Python to see how to plot this linear increasing trend using Listing 1-1. The source code for this book is available on GitHub via the book's product page, located at www.apress.com/978-1-4842-7149-0. Please note that the library imports are done once per chapter.

Listing 1-1. Getting the coffee example into Python and plotting the trend

```
import pandas as pd
import matplotlib.pyplot as plt

years = [1965, 1970, 1975, 1980, 1985, 1990, 1995, 2000, 2005, 2010, 2015, 2020]
prices = [1.00, 1.20, 1.40, 1.60, 1.80, 2.00, 2.20, 2.40, 2.60, 2.80, 3.00, 3.20]

data = pd.DataFrame({
    'year' : years,
    'prices': prices
})
ax = data.plot.line(x='year')
ax.set_title('Coffee Price Over Time', fontsize=16)
plt.show()
```

You will obtain the graph displayed in Figure 1-1.

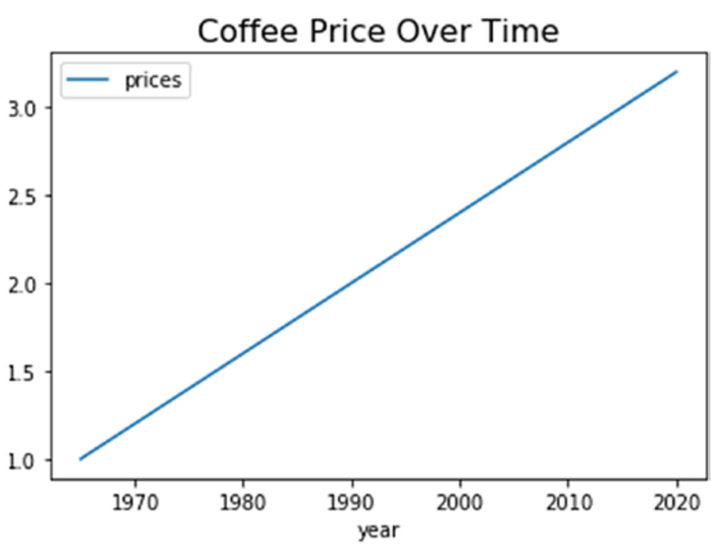

Figure 1-1. *The plot of the coffee price example*

To make predictions for the price of coffee in this hypothetical town, you could just put your ruler next to the graph and continue the upward line: the prediction for this variable does not need any **explanatory variables** other than its past values. The historical data of this example allows you to forecast the future. This is a determining characteristic of **time series models**.

Now let's see a comparable example but with the prices of hot chocolate rather than the prices of a cup of coffee and quarterly data rather than data every five years (Table 1-2).

Table 1-2. *Hot Chocolate Prices Over the Years*

Period	Average Price
Spring 2018	2.80
Summer 2018	2.60
Autumn 2018	3.00
Winter 2018	3.20
Spring 2019	2.80
Summer 2019	2.60
Autumn 2019	3.00
Winter 2019	3.20
Spring 2020	2.80
Summer 2020	2.60
Autumn 2020	3.00
Winter 2020	3.20

Do you see the trend? In the case of hot chocolate, you do not have a year-over-year increase in price, but you do detect **seasonality**: in the example, hot chocolate prices follow the temperatures of the seasons. Let's get this data into Python to see how to plot this seasonal trend (use Listing 1-2 to obtain the graph in Figure 1-2).

Listing 1-2. Getting the hot chocolate example into Python and plotting the trend

```
seasons = ["Spring 2018", "Summer 2018", "Autumn 2018", "Winter 2018",
           "Spring 2019", "Summer 2019", "Autumn 2019", "Winter 2019",
           "Spring 2020", "Summer 2020", "Autumn 2020", "Winter 2020"]
prices = [2.80, 2.60, 3.00, 3.20,
          2.80, 2.60, 3.00, 3.20,
          2.80, 2.60, 3.00, 3.20]

data = pd.DataFrame({
    'season': seasons,
    'price': prices
})

ax = data.plot.line(x='season')
ax.set_title('Hot Chocolate Price Over Time', fontsize=16)
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
plt.show()
```

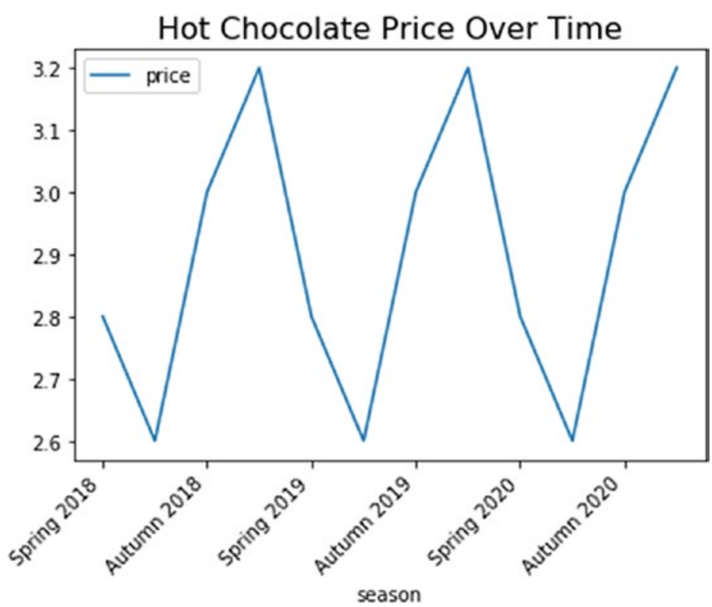

Figure 1-2. *Plot of the hot chocolate prices*

As in the previous example, you can predict the future prices of hot chocolate easily using the past data on hot chocolate prices: the prices depend only on the season and are not influenced by any explanatory variables.

Note Univariate time series models make predictions based on trends and seasonality observed in their own past and do not use explanatory variables other than the **target variable**: **the variable that you want to forecast**.

You can imagine numerous types of combinations of those two processes, for example, have both a quarterly seasonality and a linear increasing trend and so on. There are many types of processes that can be forecasted by modeling the historical values of the target variable. In Chapters 3–7, you will see numerous univariate time series models for forecasting.

Supervised Machine Learning Models

Now that you are familiar with the idea of using the past of one variable, you are going to discover a different approach to making models. You have just seen univariate time series models, which are models that use only the past of a variable itself to predict its future.

Sometimes, this approach is not logical: processes do not always follow trends and seasonality. Some predictions that you would want to make may be dependent on other, independent sources of information: **explanatory variables**.

In those cases, you can use a family of methods called **supervised machine learning** that allows you to model relationships between explanatory variables and a target variable.

A Quick Example of the Supervised Machine Learning Approach

To understand this case, you have the fictitious data in Table 1-3: a new example that contains the sales amount of a company per quarter, with three years of historical data.

Table 1-3. Quarterly Sales

Period	Quarterly Sales
Q1 2018	48,000
Q2 2018	20,000
Q3 2018	35,000
Q4 2018	32,0000
Q1 2019	16,000
Q2 2019	58,000
Q3 2019	40,000
Q4 2019	30,000
Q1 2020	32,000
Q2 2020	31,000
Q3 2020	63,000
Q4 2020	57,000

To get this data into Python, you can use the following code (Listing 1-3).

Listing 1-3. Getting the quarterly sales example into Python and plotting the trend

```
quarters = ["Q1 2018", "Q2 2018", "Q3 2018", "Q4 2018",
            "Q1 2019", "Q2 2019", "Q3 2019", "Q4 2019",
            "Q1 2020", "Q2 2020", "Q3 2020", "Q4 2020"]

sales = [48, 20, 42, 32,
         16, 58, 40, 30,
         32, 31, 53, 40]

data = pd.DataFrame({
    'quarter': quarters,
    'sales': sales
})
```

```
ax = data.plot.line(x='quarter')
ax.set_title('Sales Per Quarter', fontsize=16)
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
plt.show()
```

The graph that you obtain is a line graph that shows the sales over time (Figure 1-3).

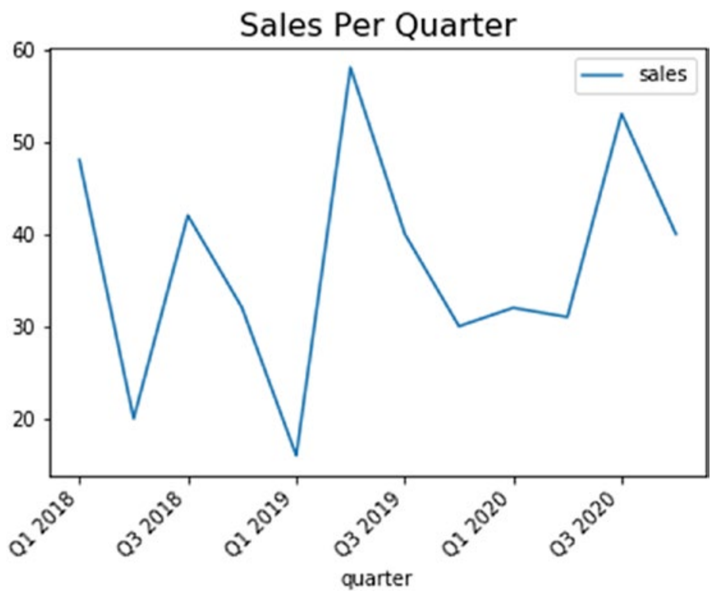

Figure 1-3. *Plot of the quarterly sales*

What you can see in this graph does not resemble the previous examples: there is no clear linear trend (neither increasing nor decreasing), and there is no clear quarterly seasonality either.

But as the data is about sales, you could imagine many factors that influence the sales that you'll realize. Let's look for explanatory variables that could help in explaining sales. In Table 1-4, the data have been updated with two explanatory variables: discount and advertising budget. Both are potential variables that could influence sales numbers.

Table 1-4. *Quarterly Sales, Discount, and Advertising Budget*

Period	Quarterly Sales	Avg. Discount	Advertising Budget
Q1 2018	48,000	4%	500
Q2 2018	20,000	2%	150
Q3 2018	35,000	3%	400
Q4 2018	32,0000	3%	300
Q1 2019	16,000	2%	100
Q2 2019	58,000	6%	500
Q3 2019	40,000	4%	380
Q4 2019	30,000	3%	280
Q1 2020	32,000	3%	290
Q2 2020	31,000	3%	315
Q3 2020	63,000	6%	625
Q4 2020	57,000	6%	585

Let's have a look at whether it would be possible to use those variables for a prediction of sales using Listing 1-4.

Listing 1-4. Getting the quarterly sales example into Python and plotting the trend

```
quarters = ["Q1 2018", "Q2 2018", "Q3 2018", "Q4 2018",
            "Q1 2019", "Q2 2019", "Q3 2019", "Q4 2019",
            "Q1 2020", "Q2 2020", "Q3 2020", "Q4 2020"]

sales = [48, 20, 42, 32,
         16, 58, 40, 30,
         32, 31, 53, 40]

discounts = [4,2,3,
             3,2,6,
             4,3,3,
             3,6,6]
```

```
advertising = [500,150,400,
               300,100,500,
               380,280,290,
               315,625,585]

data = pd.DataFrame({
    'quarter': quarters,
    'sales': sales,
    'discount': discounts,
    'advertising': advertising
})

ax = data.plot.line(x='quarter')
ax.set_title('Sales Per Quarter', fontsize=16)
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
plt.show()
```

This gives you the graph that is displayed in Figure 1-4: a graph displaying the development of the three variables over time.

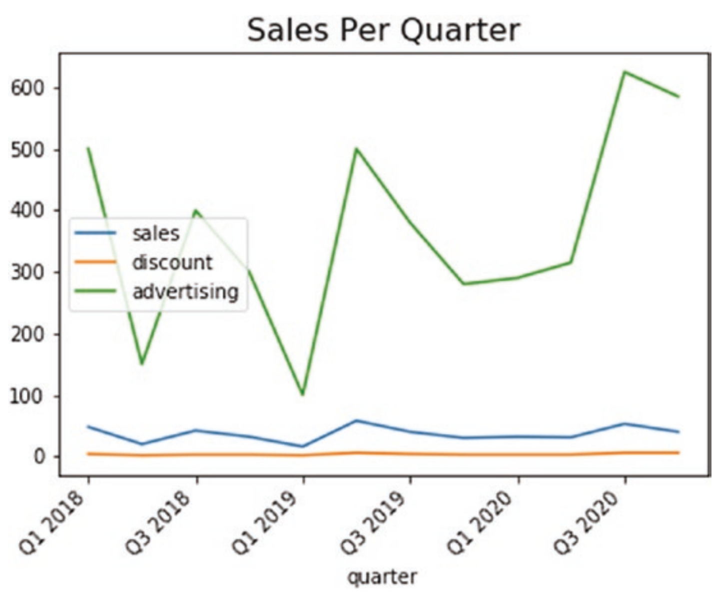

Figure 1-4. *Plot of the sales per quarter with correlated variables*

At this point, visually, you'd probably say that there is not a very important relationship between the three variables. But let's have a more zoomed-in look at the same graph (Listing 1-5).

Listing 1-5. Zooming in on the correlated variables of the quarterly sales example

```
quarters = ["Q1 2018", "Q2 2018", "Q3 2018", "Q4 2018",
            "Q1 2019", "Q2 2019", "Q3 2019", "Q4 2019",
            "Q1 2020", "Q2 2020", "Q3 2020", "Q4 2020"]

sales = [48, 20, 42, 32,
         16, 58, 40, 30,
         32, 31, 53, 40]

discounts = [4,2,3,
             3,2,6,
             4,3,3,
             3,6,6]

discounts_scale_adjusted = [x * 10 for x in discounts]

advertising = [500,150,400,
               300,100,500,
               380,280,290,
               315,625,585]

advertising_scale_adjusted = [x / 10 for x in advertising]

data = pd.DataFrame({
    'quarter': quarters,
    'sales': sales,
    'discount': discounts_scale_adjusted,
    'advertising': advertising_scale_adjusted
})

ax = data.plot.line(x='quarter')
ax.set_title('Sales Per Quarter', fontsize=16)
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
plt.show()
```

This gives the graph displayed in Figure 1-5: you can suddenly observe a very clear relationship between the three variables! The relationship was already there in the previous graph (Figure 1-4), but it was just not visually obvious due to the difference in scale of the curves.

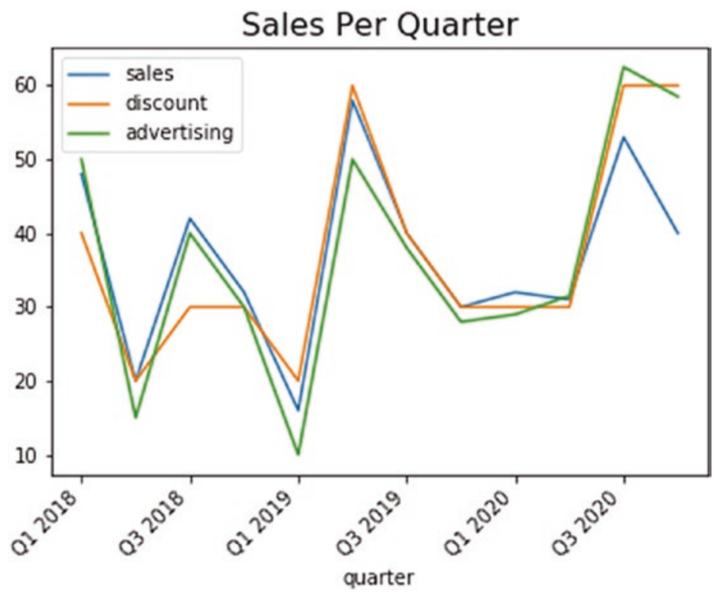

Figure 1-5. *Zoomed view of the correlated variables of the quarterly sales example*

Imagine you observe a correlation as strong as in Figure 1-5. If you had to do this sales forecast for next month, you could simply ask your colleagues what the average discount is going to be next month and what next month's advertising budget is, and you would be able to come up with a reasonable guess of the future sales.

This type of relationships is what you are generally looking at when doing supervised machine learning. Intelligent use of those relations is the fundamental idea behind the different techniques that you will see throughout this book.

Correlation Coefficient

The visual way to detect correlation is great. Yet there is a more exact way to investigate relationships between variables: the correlation coefficient. The **correlation coefficient** is a very important measure in statistics and machine learning as it determines how much two variables are correlated.

The correlation coefficient between two variables x and y can be computed as follows:

A **correlation matrix** is a matrix that contains the correlations between each pair of variables in a dataset. Use Listing 1-6 to obtain a correlation matrix.

Listing 1-6. Getting the quarterly sales example into Python and plotting the trend

```
data.corr()
```

It will give you the correlations between each pair of variables in the dataset as shown in Figure 1-6.

	sales	discount	advertising
sales	1.000000	0.848135	0.902568
discount	0.848135	1.000000	0.920958
advertising	0.902568	0.920958	1.000000

Figure 1-6. *Correlation table of the quarterly sales example*

A correlation coefficient is always **between -1 and 1**. A positive value for the correlation coefficient means that two variables are positively correlated: if one is higher, then the other is generally also higher. If the correlation coefficient is negative, there is a negative correlation: if one value is higher, then the other is generally lower. This is the **direction of the correlation**.

There is also a notion of the **strength of the correlation**. A correlation that is close to 1 or close to -1 is strong. A correlation coefficient that is close to 0 is a weak correlation. Strong correlations are generally more interesting, as an explanatory variable that strongly correlated to your variable can be used for forecasting it.