第 3 章 Chapter 3

数据分析工具 Pandas

3.1 颠覆 R 的 Pandas

进行机器学习应用的第一步是理解和探索数据，为此我们需要一套交互性很强的软件。一款理想的数据分析软件可以轻松地从多个来源读取数据、进行预处理，并且还要具有优良的统计和可视化功能，Pandas 就是这样一款软件。

Pandas 是一款基于 Python 的数据分析和建模的开源软件包。2012 年两位笔者刚刚在亚马逊相识的时候，如日中天的 R 工具正是机器学习和数据分析的主流，而基于 Python 的数据分析工具 Pandas 正在默默无闻地发展壮大。到 2016 年本书写作之时，Pandas 已经完全取代了 R，成为了主流业务中数据分析的必备软件。这样的成功与 Pandas 的设计是密不可分的。这其中有以下两个方面的原因。

- **取材于 R，超越 R：**Pandas 里处处都有 R 的影子。首先，Pandas 中数据的基本单位是 DataFrame。DataFrame 的基本概念来自于 R，其代表的是一个包含数据的基本单位。DataFrame 中的每一行代表一个观测，每一列代表一个变量，其中变量可以是数值、文本等多种类型，这样的数据结构大大方便了机器学习的准备工作。
- **优秀的生态对接：**Pandas 具有优秀的对接接口，在与文本文件、HDFS、SQL 等进行读写操作时非常方便。在可视化方面，Pandas 与 MatplotLib 可以说是整合得天衣无缝。最让人称道的是，为了向 R 致敬，Pandas 加入了一项参数，从而可以完全按照 R 的 ggplot 风格进行绘图，另外，Pandas 的底层数据结构也依赖于 Python 生态中主流的 Numpy Array，可以非常方便地调用 numpy、scipy 中已有的模块。

本章将介绍 Pandas 的基本操作。这里主要是利用 Pandas 进行初步数据清理和研究工作，我们也会对数据可视化进行初步介绍。但是对于自动化可视化呈现的工作，现今市面

上已经有了更为强大的 ELK（Elasticsearch、Logstash、Kibana）集群，该集群将在第 9 章详细介绍。

3.2 Pandas 的安装

本章节的例子存放在了官方 Github 的空间中，只需要进行以下操作即可获得所有代码和数据：

```
git clone https://github.com/real-time-machine-learning/1-pandas-intro
```

本节内容假设读者是在 Ubuntu 或 Mac 环境下进行学习的，下面的步骤可以供 Windows 用户参考，在实际操作时有可能需要稍作修改。

1. 安装 Python3

在 Ubuntu 下安装 Python3，只需执行下面的命令即可：

```
sudo apt-get install python3 python3-pip python3-dev build-essential
```

在 Mac 下利用 Homebrew 安装 Python3，只需执行下面的命令即可：

```
brew install python3
```

2. 安装 Pandas

这里通过 Python 的 Pip 配置文件来安装 Pandas。我们在后面的 Docker 学习中，将会看到这样的配置方法非常有利于自动化 Docker 操作，安装命令如下：

```
sudo pip3 install -r requirements.txt
```

如果一切顺利，上面的操作完成以后，就可以启动 Python3 并且调用 Pandas 了，命令如下：

```
python3
>>> import pandas as pd
```

3.3 利用 Pandas 分析实时股票报价数据

熟悉一项软件的最好方法就是通过示例来亲自使用它。这里将会通过分析苹果公司 2015 年 8 月 3 日秒级股票价格的数据来熟悉 Pandas 的用法。建议通过 Python 笔记本或交互式窗口的方法来进行下面的操作。

首先，需要导入相关的模块，在导入 Pandas 模块的同时，我们还用到了 Datetime 模块。Datetime 模块的主要功能是对时间、日期等数据进行处理，导入命令如下：

```
import pandas as pd
from datetime import datetime
```

3.3.1 外部数据导入

这里将会导入 2015 年 8 月 3 日苹果公司的秒级股票交易数据，不过，相应的原始数据需要稍做清理才能使用，而这正好符合本章的学习要点。

首先，用 Pandas 的 read_csv 模块直接从 csv 文件中导入数据。原始数据一共有六列，分别存有原始时间戳、每秒开盘价、最高价、最低价、收盘价和成交量信息。可以通过 names 参数将这些名字赋给处理好的数据，导入命令如下：

```
data = pd.read_csv("aapl.csv",
                   names = ["timestamp_raw","Open","High",
                            "Low","Close","Volume"],
                   index_col = False)
print(type(data))
```

上面的 type（data）可以打印出当前数据对象的类。可以看到，这里 data 对象的类名为 DataFrame，是 Pandas 中最基本的数据形态。

导入数据之后，当然还要看看我们最感兴趣的数据长什么样，在交互窗口中打印前 5 行和后 5 行。这里需要用到 DataFrame 的 head 和 tail 函数，命令如下：

```
data.head(5)
data.tail(5)
```

可以注意到记录中的股价数值为原始股价乘以 10000。

原始数据中的时间记录为每天距离格林威治标准时间的秒数乘以 1000，为增加可读性，需要将数据先还原。这里先将 data 对象的索引变为处理后的时间标记，并调用 DataFrame.index 域，示例代码如下：

```
UNIX_EPOCH = datetime(1970, 1, 1, 0, 0)
def ConvertTime(timestamp_raw, date):
    """ 该函数会将原始的时间转化为所需的 datetime 格式 """
    delta = datetime.utcfromtimestamp(timestamp_raw) - UNIX_EPOCH
    return date + delta

data.index = map(lambda x: ConvertTime(x, datetime(2015, 8, 3)),
                 data["timestamp_raw"]/1000)
```

这个时候 timestamp_raw 一列将不再有用，可以删掉它。这里调用了 DataFrame.drop( ) 函数来实现该功能：

```
data = data.drop("timestamp_raw",1)
```

3.3.2 数据分析基本操作

导入数据并做初步清理之后，可以调用 DataFrame 对象的函数对其进行各种基本的修改和描述。DataFrame 的很多操作都是通过调用对象的函数来进行的，具体有哪些函数呢？

可以通过如下 dir（ ）命令来查看：

```
dir(data)
```

经过查看可以得知，大多数的常用函数都已经包含在内了，如 mean（均值）、max（最大值）、min（最小值）。例如，为了求得该数据集每一列的均值，我们可以进行如下操作，求最大值、最小值的操作也与此类似：

```
data.mean()
```

同时还可以调用 describe 函数直接产生常用的描述性统计量，命令如下：

```
data.describe()
```

我们进行数据分析时，往往需要对数据的假设进行检验。例如美股交易时间是从美国东部时间的早上 9:30 到下午 3:30，但是很多主流股票还具有盘前和盘后交易。盘前和盘后交易时间中估价波动较大，成交量较小，对此本书不进行研讨。在进行其他分析之前，我们需要检视一下所有数据记录的时间范围。上面的统计量操作也可以在 data.index 上执行。这里 DataFrame.index 相当于一个 Series 对象，命令如下：

```
data.index.min()
data.index.max()
```

可以看到，交易时间其实包括了盘前和盘后的大量时间。在实际交易策略中，我们往往只会在正常交易时间进行交易，所以需要对数据按照时间进行拆分，只保留正常交易时间的数据，完成该项操作非常容易，命令如下：

```
data_trading_hour = data["201508030930":"201508031529"]
```

3.3.3 可视化操作

进行了简单的数据清理之后，就可以开始进行可视化操作了，首先通过目测的方式来查看数据的分布。Pandas 进行可视化操作需要依赖于 Matplolib 模块，这里首先导入对应的模块，导入命令如下：

```
import matplotlib
from matplotlib import pyplot as plt
```

Matplotlib 自带的画图风格比较僵硬，需要改改，同时为了向 R 致敬，这里设置画图风格为 R ggplot 风，设置命令如下：

```
matplotlib.style.use('ggplot')
```

画图查看每一秒的收盘价。这里只需要对 Series 类的变量调用 plot 函数，即可得到图 3-1 所示的股价走势图，调用命令如下：

```
data_trading_hour["Close"].plot()
plt.show()
```

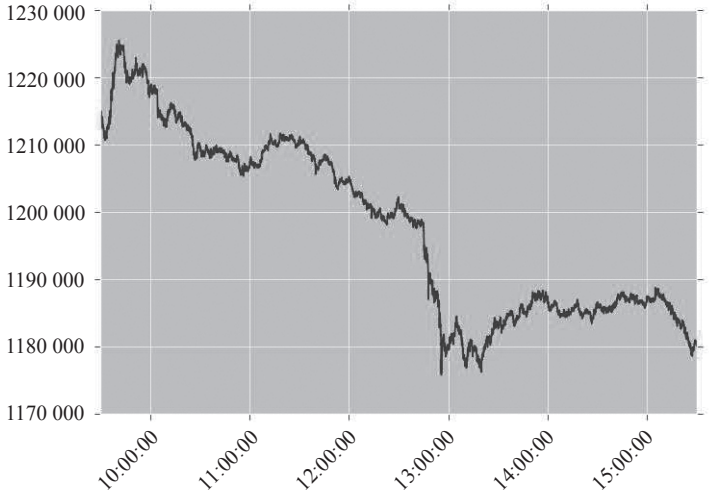

图 3-1 2015 年 8 月 3 日苹果公司股票（AAPL）股价走势（每秒收盘价）

同时，也有人可能对成交量感兴趣。根据格兰杰因果检验等研究，成交量对股价变化也有影响。每秒成交量是什么样的分布？可以通过下面的命令做出直方图。只需要调用 Series 类对象的 plot.hist 函数即可：

```
data_trading_hour["Volume"].plot.hist()
plt.show()
```

直方图画出来之后，读者将会发现大多数观测集中在了较小的范围之内，但是有若干秒的交易量是其他时候的数倍。为了更深入地研究，可以画出时序图做进一步的观察，画时序图的命令如下，得到的图形如图 3-2 所示。

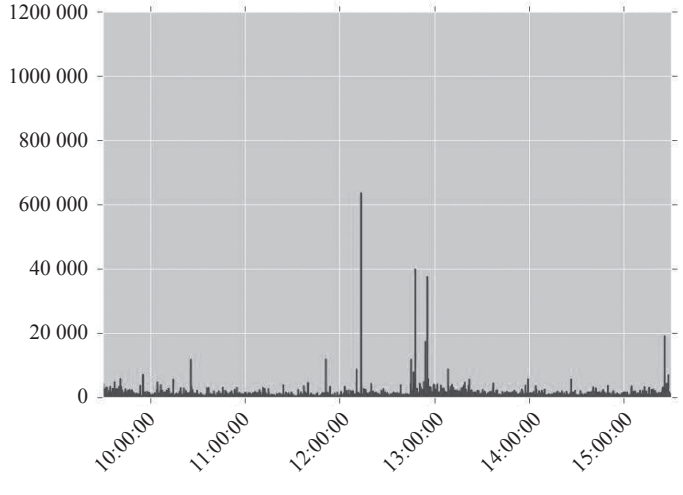

图 3-2 2015 年 8 月 3 日苹果公司股票每秒成交量可视化

```
data_trading_hour["Volume"].describe()
data_trading_hour["Volume"].plot()
plt.show()
```

果然正如我们所假设的，中午时分有大单交易发生。

3.3.4 秒级收盘价变化率初探

当然，对于实时量化交易，我们最感兴趣的还是每秒的变化率。那么下面我们就来看看股价变化的分布情况。为了到相邻时间点股价的变化率，我们可以通过调用 diff 函数来实现，得到的变化率序列也是一个 Series 类对象。就如 3.3.3 节一样，我们可以将变化率可视化，得到图 3-3。调用 diff 函数的命令如下：

```
data_trading_hour["Close"].diff().plot.hist()

plt.show()

change = data_trading_hour["Close"].diff()/data_trading_hour["Close"]
change.plot()
plt.show()
```

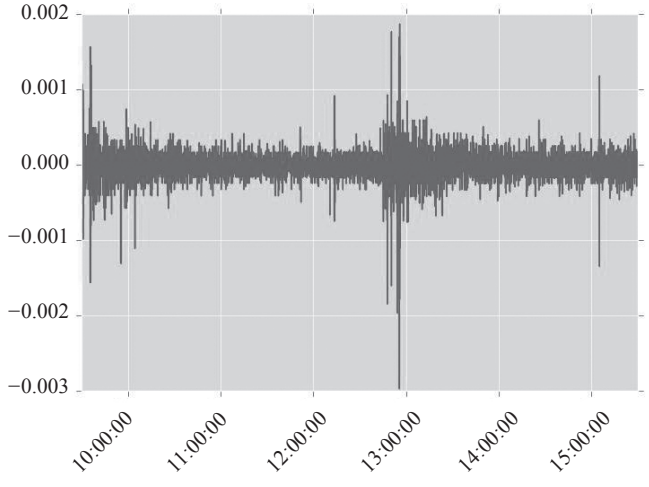

图 3-3 2015 年 8 月 3 日苹果公司股价每秒间变化率

现在回到出发点，我们分析和可视化数据是为了在后文中发掘出可能的量化交易策略。我们常常听说股价会追涨杀跌，在这种模式中的股价会按照趋势继续上涨或下跌。我们也听说过可能会均值反转，在这种模式中的股价会在具有了大幅波动之后回归平均值。那么，秒级数据又有什么样的趋势模式呢？可以通过 shift 函数对时间序列进行错位，并且通过 corr 函数计算两个时间序列之间的相关性系数。绝对值较大的相关性系数代表前后时间中股价变化的相关程度较高；绝对值近乎为 0 则代表前后时间中股票变化相关线性程度低。

shift 函数命令如下：

```
change.shift(1).corr(change)
change.shift(2).corr(change)
```

通过图 3-4 可以看到，前后一秒股价变化率的相关性系数为 −0.167，这样的相关性对于金融数据来说已经非常显著了。但是这一相关性在两秒的间隔之后迅速衰减到了 −0.034，所以这就要求我们的实时交易策略系统必须具有非常低的延迟，才能抓住这样的先机，得到超额的收益。

其实，在时间序列研究中，已经有了一套比较完备的描述性统计量，自相关性（auto-correlation）就是这样一个例子。MatplotLib 的 acorr 函数可以自动对时间序列做出自相关图，acorr 函数的命令如下：

```
plt.acorr(change[1:], lw = 2)
plt.show()
```

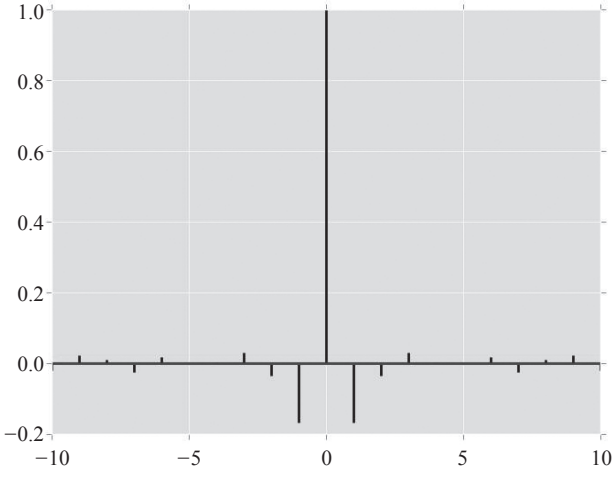

图 3-4 AAPL 股价变化率自相关系数柱状图

图 3-4 所示为 AAPL 股价变化率自相关系数柱状图，其横轴的每个刻度均代表时间序列的错位大小，1 表示时间序列与错位 1 秒的自身进行相关性计算；2 表示时间序列与错位 2 秒的自身进行相关性计算。以此类推。该图纵轴代表计算的相关性系数大小。在错位为 0 时，时间序列和自身完美相关，这里的相关性系数为 1。

从图 3-4 可以看出，苹果公司当天股价变化率的自相关性随着时间错位的增加而递减。前一秒股价变化率和后一秒股价呈负相关关系，这暗示我们在短期股票交易中，股价变化具有均值回归的模式。在均值回归模式中，如果股票出现大幅上涨或下跌，那么在后面的短时间内，可能会出现反向的股价波动，以减弱前期的变化。

3.4 数据分析的三个要点

本书后面的章节中将会以前面发现的均值回归的性质为依托，设计实时机器学习交易策略进行交易。好多读者看到这里可能已经跃跃欲试，等不及要开始搭建服务器开始赚他一个亿了。但是在这之前我们需要总结一下在开展机器学习工作前期关于数据分析的几个原则。

3.4.1 不断验证假设

验证假设是否正确是机器学习前期数据分析最重要的目的。这里的假设包括但不限于：数据的格式、变量的数量、数据是否缺失、是否有极端值、采样是否均衡等。上面这些假设，如果稍有差错，就会让在后面得到的机器学习模型无用武之地。

与此同时，我们通过数据清理得到的结果也需要经过假设验证以保证数据的完整性。最后，在实时应用中，我们往往需要考虑如下这些情况。

- **极端值**：线下建模往往都会在第一步就过滤掉极端值，但是在实时环境中，极端值是客观存在的。
- **缺失值**：再优秀的系统也有宕机出错的时候，这个时候缺失值的出现就要求系统具有灵活的错误处理能力。
- **延迟**：本章练习数据的时间戳是交易所时间，还是到达客户端服务器的时间？任何网络延迟都可能让我们的模型不再有效。多问这样的问题在进行快速机器学习应用的时候显得尤为重要。

3.4.2 全面可视化，全面监控化

为了连续验证假设，我们必须自动化数据的监控和可视化。一个完备的实时机器学习系统至少需要以下两个部件。

- 实时关键数据可视化：通过实时面板对关键数据进行可视化，让操作人员能够一目了然地判断系统和数据的健康情况。
- 实时诊断监控：通过规则设定，对异常情况进行实时判断和报警。

本书的系统架构章节（第 9 章）将介绍如何利用 ELK（Elasticsearch、Logstash、Kibana）集群实现实时数据监控。