

27

Statistical Modeling Using Statsmodels

Statsmodels 统计模型

简介线性回归、主成分分析、概率密度估计



教育点燃火焰，绝非填鸭灌输。

Education is the kindling of a flame, not the filling of a vessel.

—— 苏格拉底 (Socrates) | 古希腊哲学家 | 470 ~ 399 BC



```
statsmodels.api.nonparametric.KDEUnivariate() 构造一元 KDE
statsmodels.graphics.boxplots.violinplot() 小提琴图
statsmodels.graphics.gofplots.qqplot() QQ 图
statsmodels.graphics.plot_grid.scatter_ellipse() 散点椭圆
statsmodels.multivariate.factor.Factor() 因子分析
statsmodels.multivariate.pca.PCA() 主成分分析
statsmodels.nonparametric.kde.KDEUnivariate() 单变量核密度估计
statsmodels.nonparametric.kernel_density.KDEMultivariate() 构造多元 KDE
statsmodels.regression.linear_model.OLS() OLS 线性回归
statsmodels.regression.linear_model.WLS() 加权 OLS 线性回归
statsmodels.regression.rolling.RollingOLS() 移动 OLS 线性回归
statsmodels.tsa.ar_model.AutoReg() AR 模型
statsmodels.tsa.arima.model.ARIMA() ARIMA 模型
statsmodels.tsa.seasonal.seasonal_decompose() 季节性分解
```



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

27.1 什么是 Statsmodels?

Statsmodels 是一个 Python 库，用于估计统计模型并进行统计数据分析。在机器学习领域，Statsmodels 虽然没有像 scikit-learn 这样的机器学习库那么全面，但是 Statsmodels 提供了许多统计方法和模型，用于探索数据、进行假设检验、进行预测和模型拟合等。

Statsmodels 主要用于以下任务。

- ▶ 最小二乘线性回归 (Ordinary Least Square Regression)，用于拟合线性模型和探索线性关系。
- ▶ 方差分析 (Analysis of Variance, ANOVA)，用于比较多个组之间的差异。
- ▶ 主成分分析 (Principal Component Analysis, PCA)。
- ▶ 时间序列分析，如 ARIMA 模型。
- ▶ 非参数方法 (Nonparametric Methods)，比如核密度估计 (Kernel Density Estimation, KDE)。
- ▶ 统计假设检验 (statistical hypothesis testing)。
- ▶ 分位图，又称 QQ 图 (Quantile-Quantile plot)。

本章介绍如何使用 Statsmodels 中几个常见函数。

表 1. Statsmodels 常用模块以及示例函数

模块	描述	举例
statsmodels.graphics	统计绘图	statsmodels.graphics.boxplots.violinplot() 小提琴图 statsmodels.graphics.plot_grids.scatter_ellipse() 散点椭圆 statsmodels.graphics.gofplots.qqplot() QQ 图
statsmodels.multivariate	多元统计	statsmodels.multivariate.pca.PCA() 主成分分析 statsmodels.multivariate.factor.Factor() 因子分析
statsmodels.regression	回归分析	statsmodels.regression.linear_model.OLS() OLS 线性回归 statsmodels.regression.rolling.RollingOLS() 移动 OLS 线性回归 statsmodels.regression.linear_model.WLS() 加权 OLS 线性回归
statsmodels.nonparametric	非参数方法	statsmodels.nonparametric.kde.KDEUnivariate() 单变量核密度估计
statsmodels.tsa	时间序列	statsmodels.tsa.ar_model.AutoReg() AR 模型 statsmodels.tsa.arima.model.ARIMA() ARIMA 模型 statsmodels.tsa.seasonal.seasonal_decompose() 季节性分解

27.2 平面散点图 + 椭圆

上一章在介绍高斯分布时，我们知道了二元高斯分布和椭圆的关系。

平面散点图 (scatter plot) 是一种常用的可视化方式，用于展示两个变量之间的关系。它将各个数据点表示为笛卡尔坐标系上的点。scatter_ellipse 函数是 statsmodels.graphics.plot_grids 模块的一部分，用于创建带有椭圆表示置信区间的散点图。简单来说，scatter_ellipse 函数在基本散点图的基础上添加了椭圆，用于展示样本数据的置信区间。

图 1 所示为鸢尾花数据的“平面散点图 + 椭圆”。图 2、图 3、图 4 这三幅图考虑了鸢尾花标签。

注意，`scatter_ellipse` 函数默认图像线条颜色为黑色。图 1 ~ 图 4 这四幅图在后期处理时修改了颜色。此外，图中下三角相关性系数矩阵热图来自本书第 23 章。

鸢尾花书《统计至简》第 23 章将会介绍这四幅图背后的数学工具。

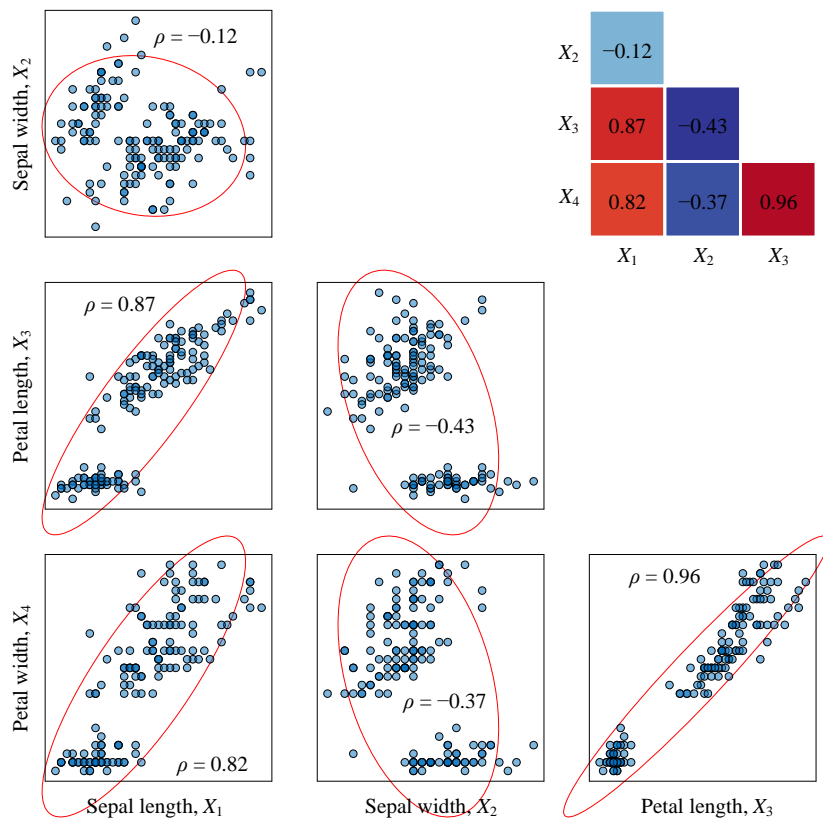


图 1. 平面散点图 + 椭圆，鸢尾花数据集

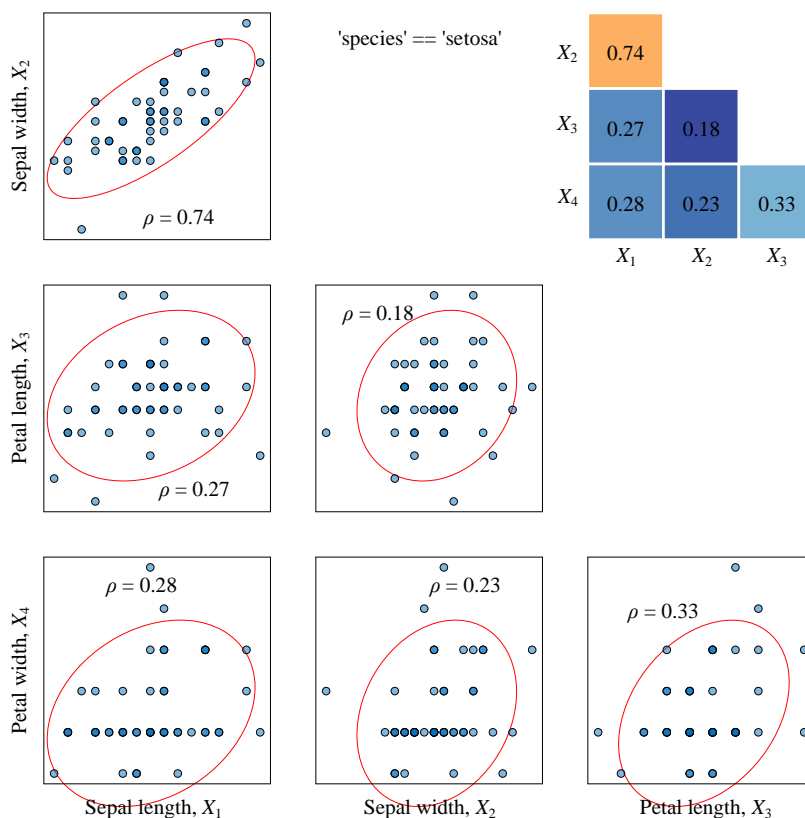


图 2. 平面散点图 + 椭圆, 鸢尾花数据集, 'species' == 'setosa'

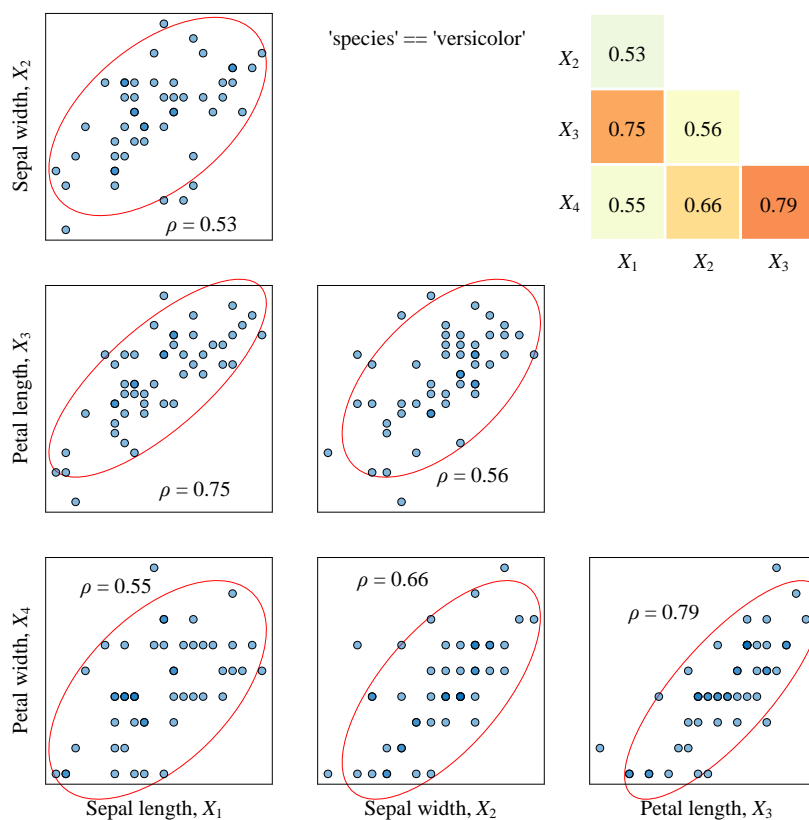


图 3. 平面散点图 + 椭圆, 鸢尾花数据集, 'species' == 'versicolor'

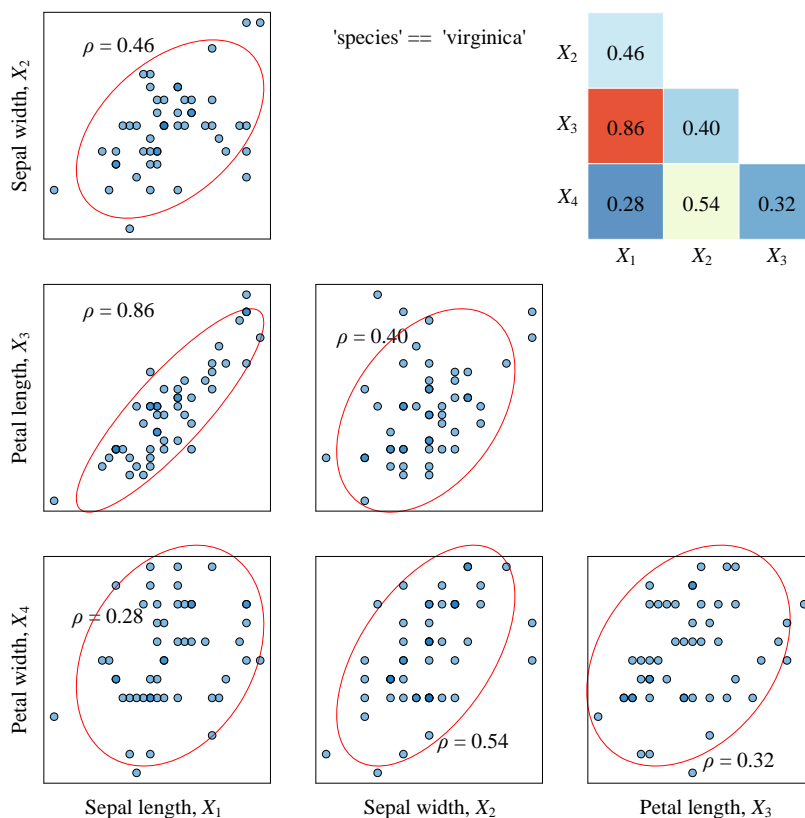


图 4. 平面散点图 + 椭圆, 鸢尾花数据集, 'species' == 'virginica'

图 5 所示为绘制图 1 ~ 图 4 的代码。

- a** 从 statsmodels 库的 plot_grids 模块中访问 scatter_ellipse 函数。
- b** 绘制平面散点图 + 椭圆。scatter_ellipse 函数中, level (默认 0.9) 是一个可选参数, 用于控制绘制椭圆时表示置信区间的置信水平 (confidence level)。置信区间是一个范围, 用于表示对一个未知参数的估计。一个 95% 的置信区间意味着我们有 95% 的置信度认为真实的参数值位于该区间内。
- c** 用 loc 选取鸢尾花不同标签样本数据。


```

import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
a from statsmodels.graphics.plot_grids import scatter_ellipse
# 导入鸢尾花数据
data_raw = sns.load_dataset('iris')
labels = ['Sepal length', 'Sepal width',
          'Petal length', 'Petal width']

fig = plt.figure(figsize=(8,8))
b scatter_ellipse(data_raw.iloc[:, :-1],
                  varnames=labels, fig=fig)
fig.savefig('散点 + 椭圆.svg', format='svg')

c for s_idx in data_raw.species.unique():
    data = data_raw.loc[data_raw.species == s_idx].iloc[:, :-1]
    fig = plt.figure(figsize=(8,8))
    scatter_ellipse(data, varnames=labels, fig=fig)
    fig.savefig('散点 + 椭圆 ' + s_idx + '.svg', format='svg')

```

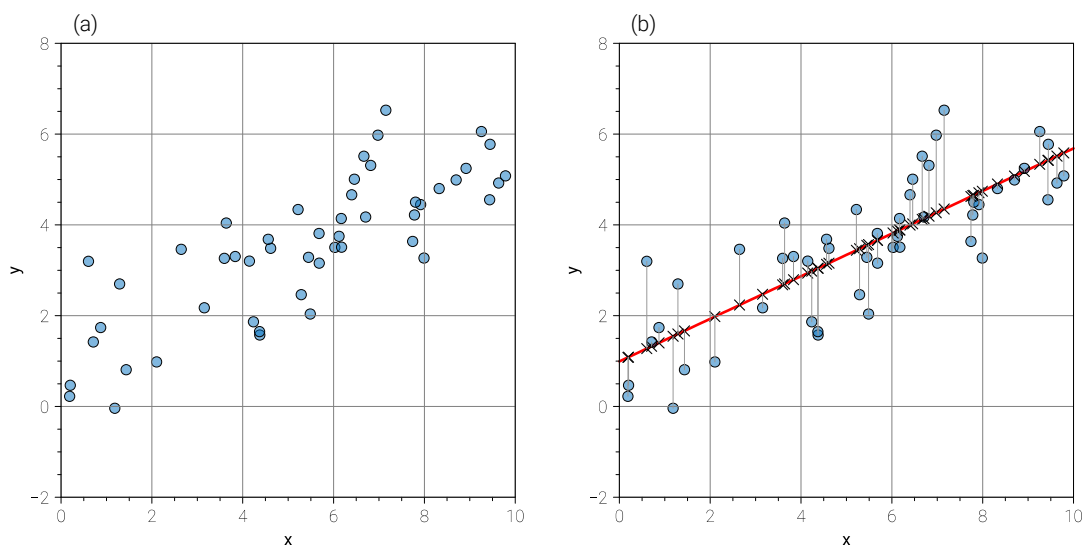
图 5. 平面散点图 + 椭圆; 

27.3 最小二乘线性回归

最小二乘 (Ordinary Least Square, OLS) 线性回归 (linear regression) 是一种用于建立线性模型的统计学方法，其目标是通过找到最佳拟合直线来预测因变量和一个或多个自变量之间的线性关系。这种方法被广泛应用于各种领域，包括数据分析、机器学习等等。

如图 6 (a) 所示，在最小二乘线性回归中，我们尝试找到一条直线，使得所有数据点到这条线的距离之和最小。这里的“距离”通常是指因变量与回归线预测值之间的差异，称为残差。图 6 (b) 中灰色线段就是残差。

我们的目标是最小化所有数据点的残差平方和，因此称为“最小二乘”。



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

图 6. 一元线性回归

图 7 代码绘制图 6 (b)。

- a 产生用于回归的样本数据。
 - b 中 `sm.add_constant(x_data)` 是 `statsmodels` 中的一个函数，用于在矩阵或数组 `x_data` 的左侧添加全 1 常数项，目的是为了计算截距项。
 - c 进行最小二乘线性回归分析。
 - d 调用 `fit()` 方法来对模型进行拟合，从而得到对应的回归系数和其他相关统计信息。
 - e 打印回归结果，具体如图 8 所示。
- 鸢尾花书《数据有道》将逐一介绍图 8 这些回归分析结果。
- f 中 `results.params` 保存线性回归结果，`results.params[1]` 为斜率 b_1 ，`results.params[0]` 为截距 b_0 。一元线性回归的解析式为 $y = b_1x + b_0$ 。
 - g 绘制预测值 (predicted value) 散点图，图 6 (b) 中的 \times 。图 6 (b) 中的蓝色点 \bullet 为样本数据。
 - h 绘制样本值 \bullet 和预测值 \times 连线线段。这个线段代表误差。

本书第 30 章还会继续介绍 Scikit-Learn 中的回归算法工具。

```

import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt

# 生成随机数据
num = 50
np.random.seed(0)
a x_data = np.random.uniform(0,10,num)
y_data = 0.5 * x_data + 1 + np.random.normal(0, 1, num)
data = np.column_stack([x_data,y_data])

# 添加常数列
b X = sm.add_constant(x_data)
# 创建一元OLS线性回归模型
c model = sm.OLS(y_data, X)
# 拟合模型
d results = model.fit()
# 打印回归结果
e print(results.summary())
# 预测
f x_array = np.linspace(0,10,101)
predicted = results.params[1] * x_array + results.params[0]

fig, ax = plt.subplots()
ax.scatter(x_data, y_data)
g ax.scatter(x_data, results.fittedvalues,
            color = 'k', marker = 'x')
ax.plot(x_array, predicted,
        color = 'r')

data_ = np.column_stack([x_data,results.fittedvalues])

h ax.plot([[i for (i,j) in data_], [i for (i,j) in data_]],
          [[j for (i,j) in data_], [j for (i,j) in data_]],
          c=[0.6,0.6,0.6], alpha = 0.5)

ax.set_xlabel('x'); ax.set_ylabel('y')
ax.set_aspect('equal', adjustable='box')
ax.set_xlim(0,10); ax.set_ylim(-2,8)
fig.savefig('一元线性回归.svg', format='svg')

```

图 7. 一元 OLS 线性回归;

```

=====
OLS Regression Results
=====
Dep. Variable:          y          R-squared:          0.656
Model:                OLS        Adj. R-squared:       0.649
Method:             Least Squares  F-statistic:        91.59
Date:               XXXXXXXXXXXX  Prob (F-statistic):  1.05e-12
Time:               XXXXXXXXXX   Log-Likelihood:     -67.046
No. Observations:    50          AIC:                138.1
Df Residuals:        48          BIC:                141.9

Covariance Type: nonrobust

=====
              coef      std err          t      P>|t|      [0.025      0.975]
-----
const          0.9928      0.296      3.358      0.002      0.398      1.587
x1             0.4693      0.049     9.570      0.000      0.371      0.568
=====
Omnibus:                 1.199   Durbin-Watson:       2.274
Prob(Omnibus):           0.549   Jarque-Bera (JB):    1.213
Skew:                    0.283   Prob(JB):            0.545
Kurtosis:                 2.487   Cond. No.            13.6
=====

```

图 8. 一元 OLS 线性回归结果

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

27.4 主成分分析

主成分分析 (principal component analysis, PCA) 是数据降维的重要方法之一。简单来说，通过线性变换，主成分分析将原始多维数据投影到一个新的正交坐标系，将原始数据中的最大方差成分提取出来。

举个例子，主成分分析实际上寻找数据在主元空间内投影。图 9 所示杯子，它是一个 3D 物体，在一张图展示杯子，而且尽可能多地展示杯子细节，就需要从空间多个角度观察杯子并找到合适角度。这个过程实际上是将三维数据投影到二维平面过程。这也是一个降维过程，即从三维变成二维。图 10 展示杯子六个平面上投影结果。

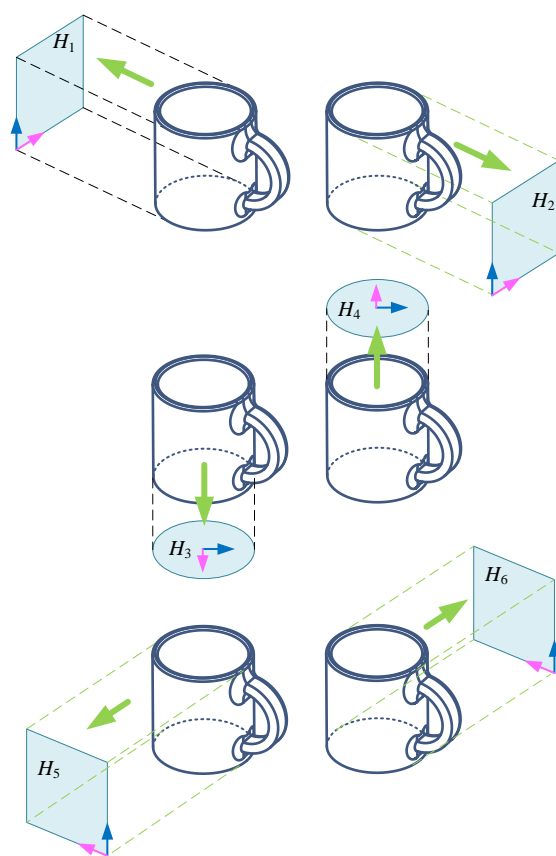


图 9. 咖啡杯六个投影方向

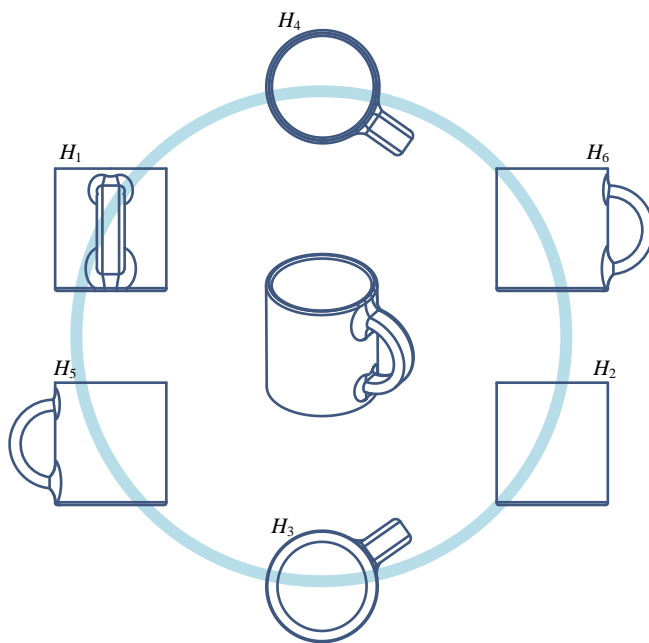


图 10. 咖啡杯在六个方向投影图像

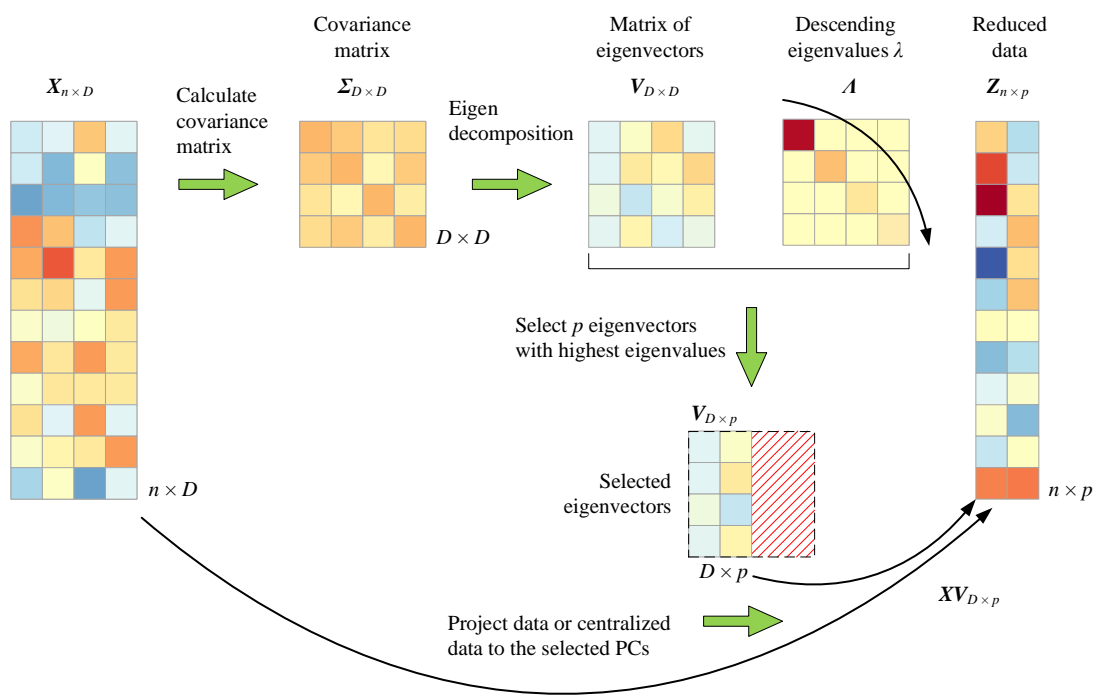


图 11. 主成分分析一般技术路线：特征值分解协方差矩阵

如图 11 所示，PCA 的一般步骤如下：

- ▶ 计算原始数据 $X_{n \times D}$ 的协方差矩阵 $\Sigma_{D \times D}$ ；
- ▶ 对 Σ 特征值分解，获得特征值 λ_i 与特征向量矩阵 $V_{D \times D}$ ；

- ◀ 对特征值 λ_i 从大到小排序，选择其中特征值最大的 p 个特征向量；
- ◀ 将原始数据（中心化数据）投影到这 p 个正交向量构建的低维空间中，获得得分 $Z_n \times p$ 。

很多时候，在第一步中，我们先**标准化**（standardization）原始数据，即计算 X 的 Z 分数。标准化防止不同特征上方差差异过大。而有些情况，对原始数据 $X_n \times D$ 进行中心化（去均值）就足够了，即将数据质心移到原点。

下面，我们用不同年期利率时间序列数据介绍如何使用 Statsmodels 函数完成主成分分析。图 12 所示为 2022 年 8 个不同年期利率走势，也就是说数据有 8 个特征（维度）。

我们先看一下图 16 代码。我们在本书前文已经介绍过 **a** ~ **f**，请大家回顾这些代码的作用。

g 用 `seaborn.lineplot()` 绘制利率走势线图。

h 用 `pct_change()` 计算日收益率。如图 13 所示，日收益率是用来衡量股票、利率在一天内的价格变动幅度的指标。日收益率通常以百分比形式表示，计算方法为：日收益率 = (当日收盘价 - 前一日收盘价) / 前一日收盘价 × 100%。日收益率数据 X 是下文主成分分析对象。

i 用 `seaborn.pairplot()` 绘制成对散点图，用来理解变量之间的关系和分布情况。对角线上的子图默认是每个变量的直方图，图 14 将对角线子图修改为概率密度估计线图，这是下一节要介绍的内容。非对角线上的图形是变量之间的散点图，图 14 仅仅保留了下三角部分子图。

j 计算日收益率数据 X 相关性系数矩阵。**k** 用 `seaborn.heatmap()` 可视化相关性系数矩阵。

如图 14 所示，从时间序列的涨跌，我们可以看到明显的联动性（co-movement）。图 15 所示的相关性系数矩阵则“量化”联动性。主成分分析 PCA 便可以帮助我们分析这种联动性。

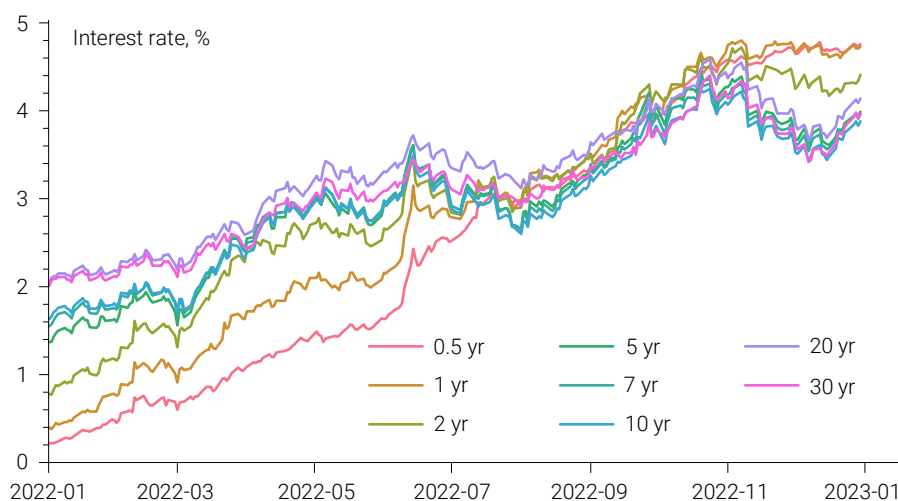


图 12. 不同年期利率时间序列数据

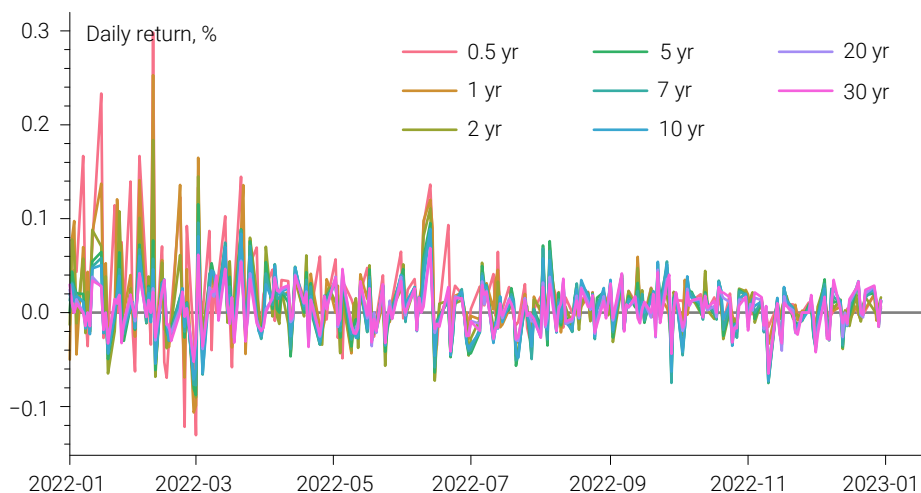


图 13. 不同年期利率日收益率时间序列数据

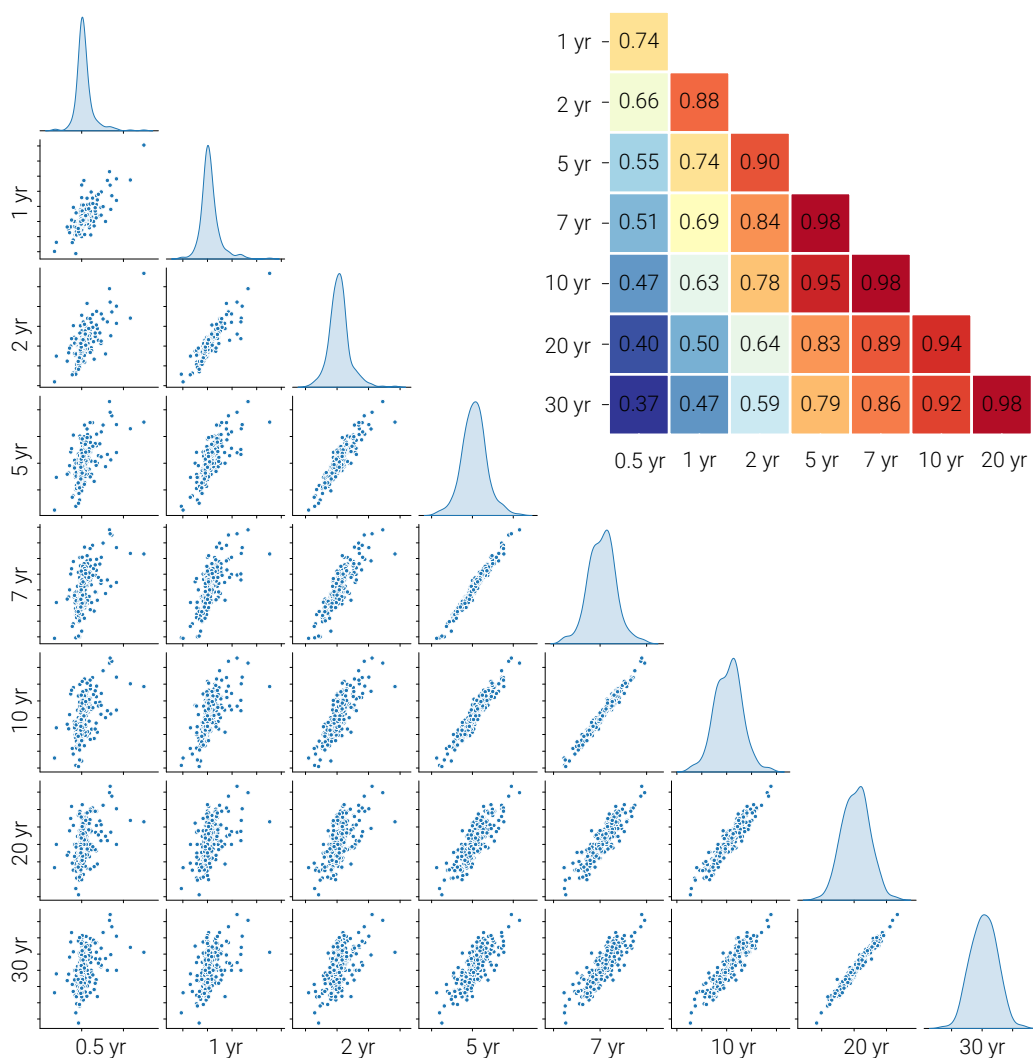


图 14. 成对特征散点图, 下三角

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

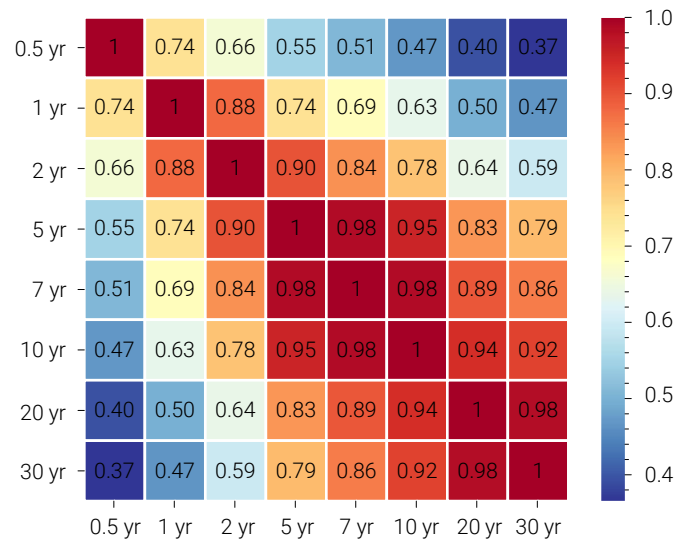


图 15. 相关性系数矩阵

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
a import pandas_datareader as pdr
b # pip install pandas_datareader
import seaborn as sns
c import statsmodels.multivariate.pca as pca
# 下载数据
d df = pdr.data.DataReader(['DGS6M0', 'DGS1',
                           'DGS2', 'DGS5',
                           'DGS7', 'DGS10',
                           'DGS20', 'DGS30'],
                           data_source='fred',
                           start='01-01-2022',
                           end='12-31-2022')

e df = df.dropna()
# 修改数据帧列标签
f df = df.rename(columns={'DGS6M0': '0.5 yr',
                           'DGS1': '1 yr',
                           'DGS2': '2 yr',
                           'DGS5': '5 yr',
                           'DGS7': '7 yr',
                           'DGS10': '10 yr',
                           'DGS20': '20 yr',
                           'DGS30': '30 yr'})

# 绘制利率走势
fig, ax = plt.subplots(figsize = (6,3))
g sns.lineplot(df, markers=False, dashes=False,
               palette = "husl", ax = ax)
ax.legend(loc='lower right', ncol=3)
# 计算日收益率
h X_df = df.pct_change()
X_df = X_df.dropna()
# 可视化收益率
fig, ax = plt.subplots(figsize = (6,3))
sns.lineplot(X_df, markers=False,
             dashes=False, palette = "husl", ax = ax)
ax.legend(loc='upper right', ncol=3)
# 成对特征散点图
i sns.pairplot(X_df, corner=True, diag_kind="kde")
# 相关性系数矩阵
j C = X_df.corr()
fig, ax = plt.subplots()
k sns.heatmap(C, ax = ax,
              annot=True,
              cmap = 'RdYlBu_r',
              square = True)

```


图 16. 下载分析利率数据; 

图 17 所示的陡坡图 (scree plot) 是 PCA 重要的可视化方案, 用于帮助确定保留多少主成分。

首先, 将原始数据进行主成分分析, 计算出各个主成分及其对应的特征值, 方差解释比例。

然后, 将每个主成分的特征值绘制在一个陡坡图上 (图 17 左纵轴)。横轴表示主成分的序号, 纵轴表示对应的特征值。

通常，特征值会从小到大排列。观察陡坡图，寻找特征值开始急剧下降的拐点。这些拐点所对应的主成分通常是数据中最重要的部分，包含了最多的信息。拐点之后的主成分的贡献较小，可以考虑不予保留。

此外，我们还可以通过量化方法来决定保留主成分的数量。

图 17 右纵轴展示累积解释总方差百分比。我们可以发现，前 3 个主成解释超过 95% 的方差。这样做可以在保留重要信息的同时降低数据的维度。也就是说，利用主成分分析，我们可以把 8 个维度降到 3 个维度，并尽可能保证数据的重要信息。

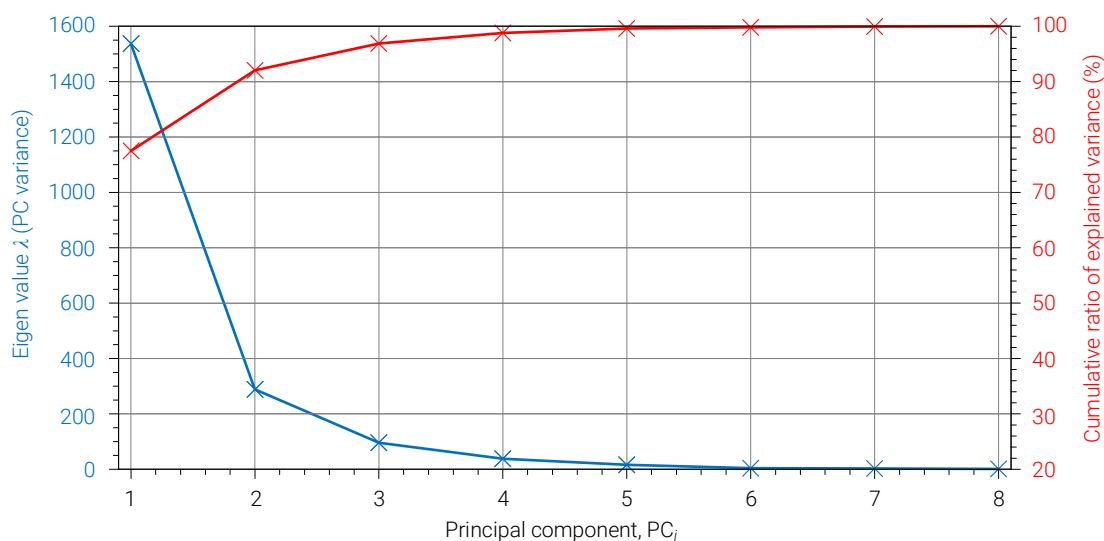


图 17. 陡坡图

在主成分分析中，载荷（loadings）是一个重要的概念，用于表示原始数据特征与各个主成分之间的线性关系。载荷反映了原始数据在每个主成分上的投影权重，从而帮助我们理解主成分的含义和解释。

具体来说，对于每个主成分，都有一组载荷值与之对应。图 18 所示为前 3 主成分载荷。

这些载荷值构成了一个向量，表示了原始特征在主成分上的投影权重。载荷值可以为正或负，它们的绝对值越大，表示该主成分与对应特征之间的关系越强。

在 PCA 的过程中，主成分的计算涉及到特征值分解数据的协方差矩阵 Σ , $\Sigma = V\Lambda V^T$ 。从数学角度来看，载荷本质上就是 V 。

鸢尾花书《矩阵力量》第 13、14 章将专门介绍特征值分解。

在主成分分析中，主成分得分（principal component score）是指原始数据在降维后的主成分空间中的投影值。如图 19 所示，主成分分数是在进行数据降维后，将原始数据点映射到新的主成分空间中的一种表示。

如图 20 所示，每个主成分都是原始特征的线性组合。大家可以自行计算所有主成分得分的相关性系数矩阵，容易发现这个矩阵为单位阵。

由于我们仅仅保留 3 个主成分，图 20 便代表降维（8 维到 3 维）过程。

注意，虽然主成分分析和线性回归都使用线性模型，但它们的目的是和使用方式不同。

主成分分析是用于降维的一种无监督学习方法，目的是找到一组新的变量，使得这些变量能够最大程度地解释原始数据中的方差。这些新的变量称为主成分，它们是原始数据中所有变量的线性组合。主成分分析通常用于数据探索和可视化，以及在高维数据中寻找最重要的特征。

而线性回归是用于预测的一种有监督学习方法，目的是通过拟合一个线性函数来预测一个连续的目标变量。线性回归通常用于建立输入变量和输出变量之间的关系，并用于预测新的输出变量值。

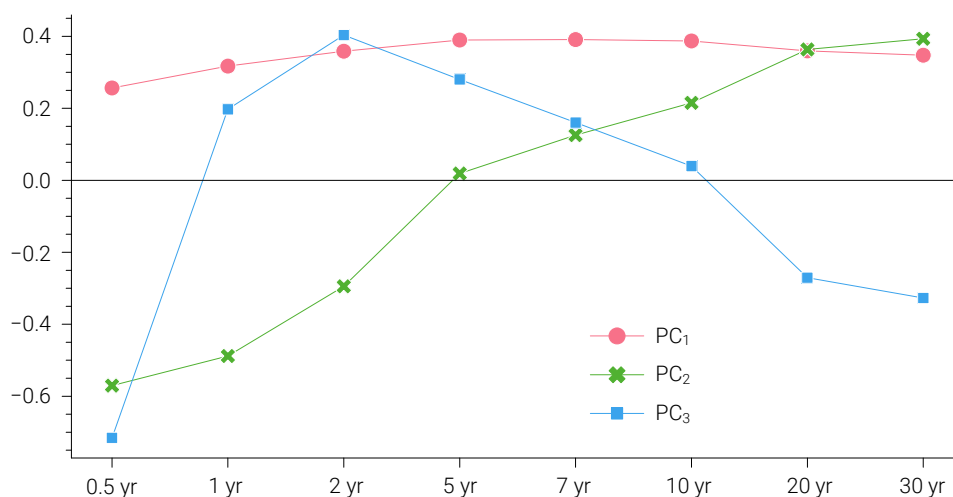


图 18. 前三主成分载荷

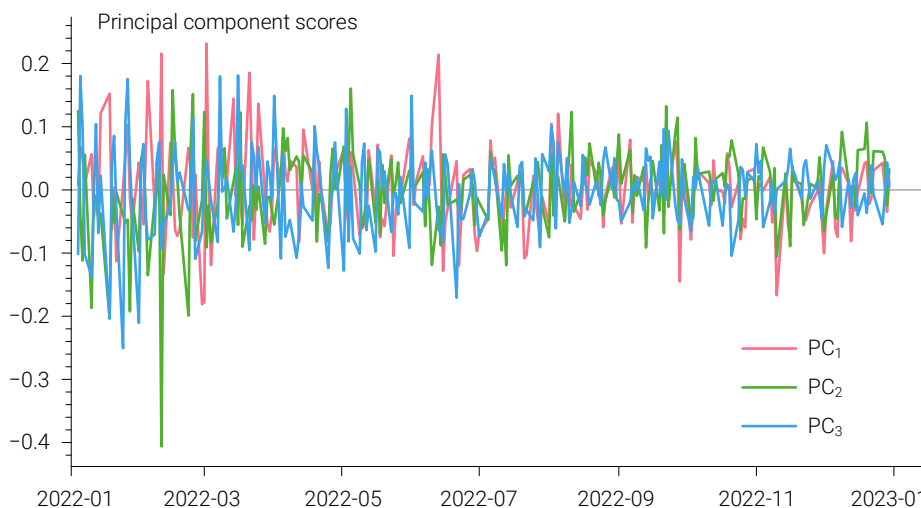


图 19. 前三主成分得分

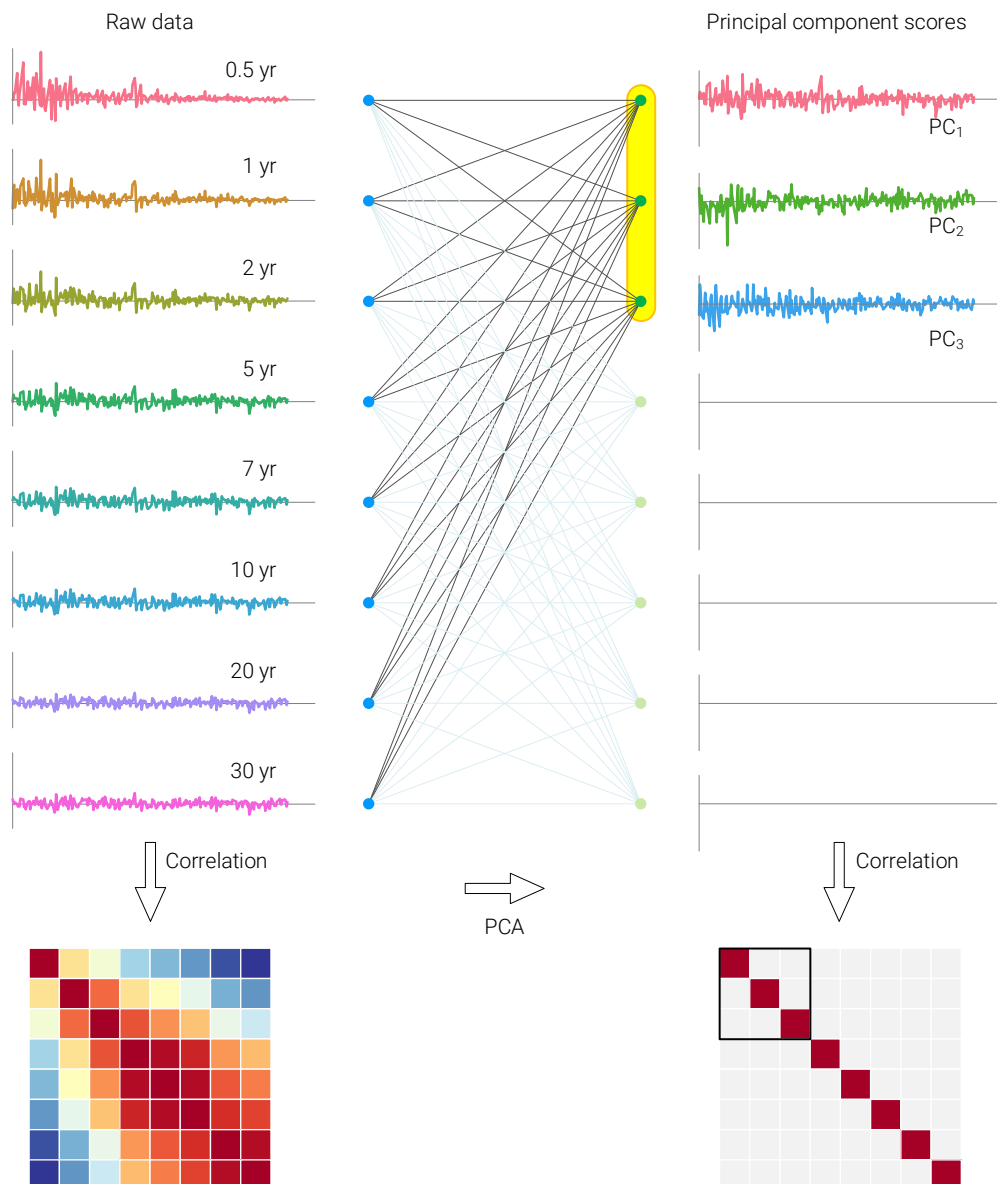


图 20. 从原始数据到主成分得分

如图 21 所示，我们用三组主成分分析“还原”原始数据，得到的结果我们称之为还原数据。这个过程实际上将主成分分数反向投影到原始数据空间。

在 PCA 中，我们通过将原始数据投影到主成分上得到主成分分数。而将主成分分数反向投影回原始数据空间，得到的数据就是还原数据（approximated data）。

投影数据与原始数据的关系是，通过主成分分析的投影过程，将原始数据映射到主成分空间，并且反向投影过程可以近似地重构出原始数据。

然而，由于 PCA 是一种降维技术，反向投影得到的数据会在重构过程中损失一些细节信息，因此反向投影出的数据可能与原始数据存在差异。图 22 和图 23 分别用散点图、线图可视化原始数据、还原数据、误差。

接着图 16 代码，图 24 代码完成主成分分析。

a 利用 `statsmodels.multivariate.pca.PCA()` 完成主成分分析。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

下面简单介绍这个函数的关键参数。`ncomp` 指定返回主成分数量，默认返回和原数据特征数一致的主成分数量。`standardize` 指定是否标准化数据，如果 `standardize = True` 相当于对原始数据相关性系数矩阵进行特征值分解，来完成主成分分析运算。`demean` 指定是否去均值，如果 `standardize = True`，默认数据已经去均值。`method = 'svd'`（默认）代表利用奇异值分解进行主成分分解，`method = 'eig'` 代表利用特征值分解完成 PCA。

b 提取特征值，从大到小排列。特征值分解将协方差矩阵转化为一组特征向量和特征值。这些特征值排列从大到小的意义在于决定了主成分的重要性和解释力。

主成分分析的目标之一是将原始数据映射到一组新的主成分上，这些主成分按照重要性递减排列。换句话说，通过选择前几个特征值较大的主成分，我们能够保留大部分原始数据的方差信息，同时实现数据的降维。这有助于更好地理解数据的结构、模式。

c 增加双 y 轴的右侧纵轴对象。

d 提取前 3 主成分。从特征值分解结果来看，这三个主成分对应的特征值分别约为 1537、288、95。三者之和占总特征值超过 95%。

e 用前 3 主成分创建还原数据。

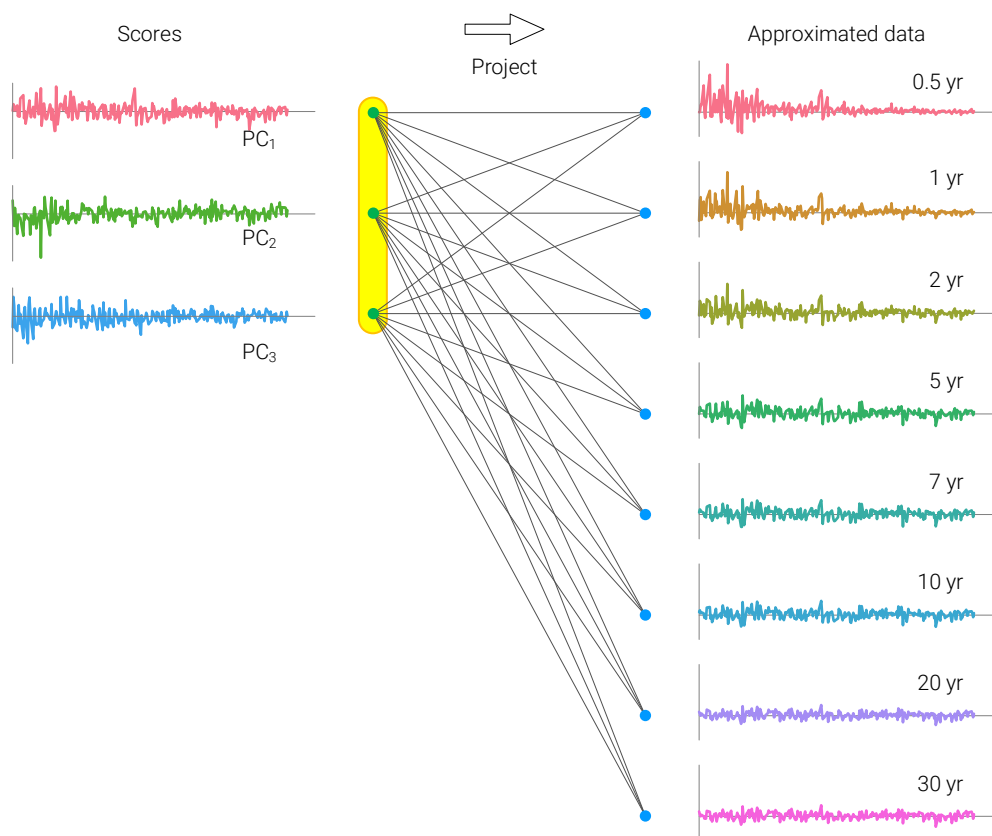


图 21. 从主成分得分（前 3 个主成分）到还原数据

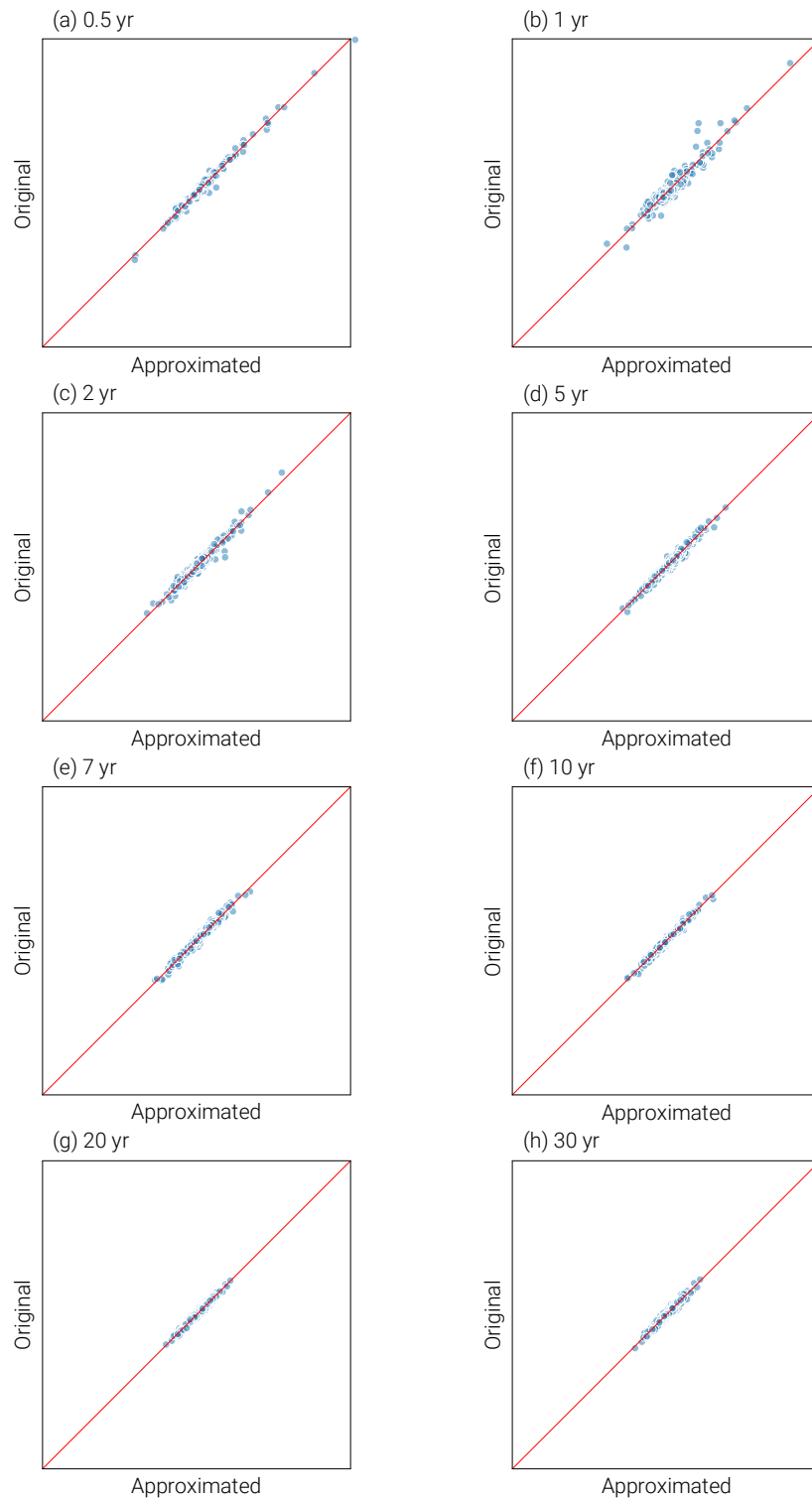


图 22. 比较原始数据和还原数据（前三主成分还原），散点图

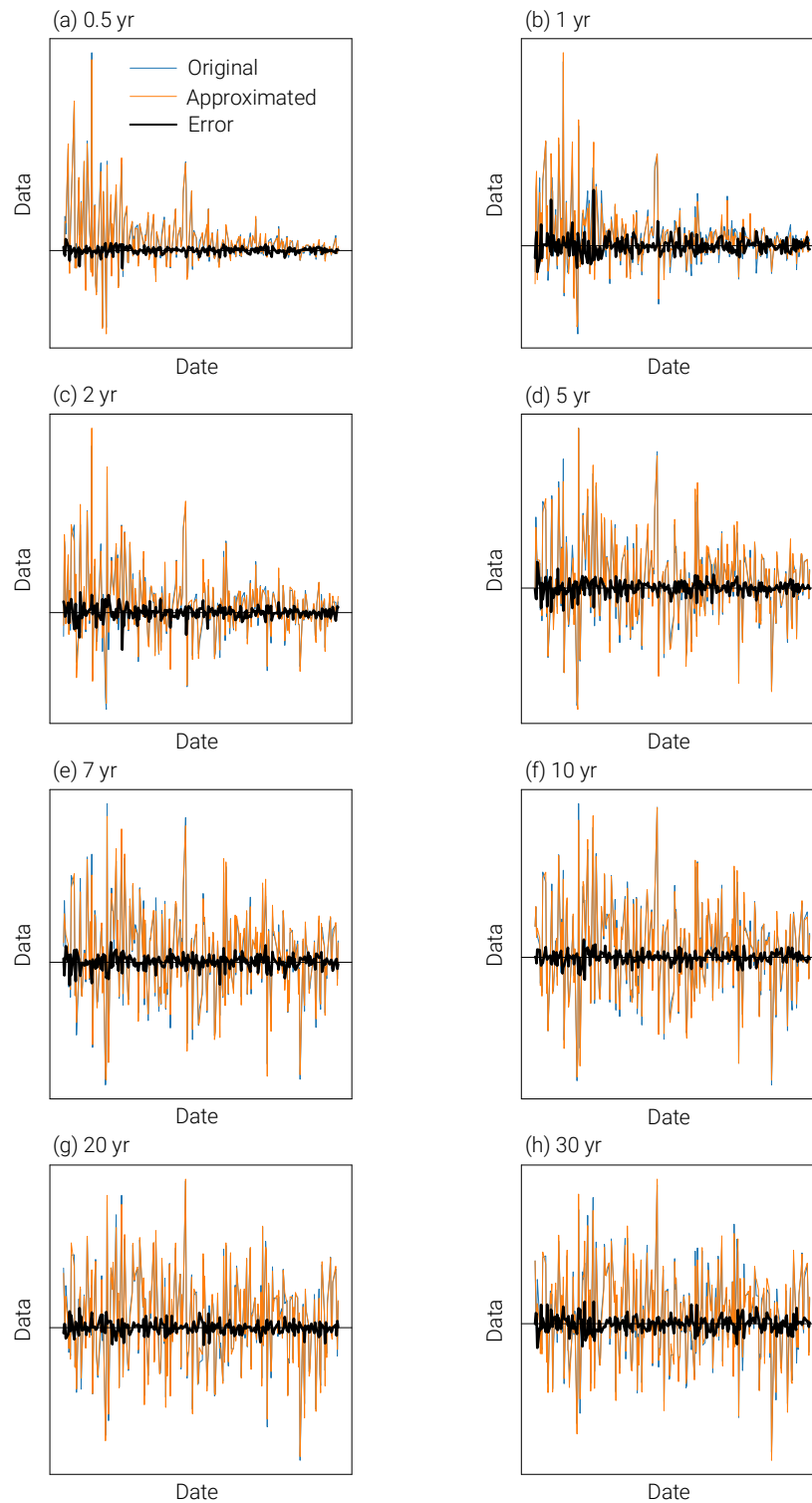


图 23. 比较原始数据和还原数据（前三主成分还原），线图

```

# 主成分分析
a pca_model = pca.PCA(X_df, standardize=True)
b variance_V = pca_model.eigenvals
# 计算主成分的方差解释比例
explained_var_ratio = variance_V / variance_V.sum()
PC_range = np.arange(len(variance_V)) + 1
labels = ['$PC_' + str(index) + '$' for index in PC_range]
# 陡坡图
fig, ax1 = plt.subplots(figsize = (6,3))

ax1.plot(PC_range, variance_V, 'b', marker = 'x')
ax1.set_xlabel('Principal Component')
ax1.set_ylabel('Eigen value $\lambda$ (PC variance)',
color='b')
ax1.set_ylim(0,1600); ax1.set_xticks(PC_range)

c ax2 = ax1.twinx()
ax2.plot(PC_range, np.cumsum(explained_var_ratio)*100,
'r', marker = 'x')
ax2.set_ylabel('Cumulative ratio of explained variance (%)',
color='r')
ax2.set_ylim(20,100)
ax2.set_xlim(PC_range.min() - 0.1, PC_range.max() + 0.1)
# PCA载荷
d loadings= pca_model.loadings[['comp_0', 'comp_1', 'comp_2']]
fig, ax = plt.subplots(figsize = (6,4))
sns.lineplot(data=loadings,
markers=True, dashes=False, palette = "husl")
plt.axhline(y=0, color='r', linestyle='-')
# 用前3主成分获得还原数据
e X_df_ = pca_model.project(3)
# 比较原始数据和还原数据
# 线图
fig, axes = plt.subplots(4,2,figsize=(4,8))
axes = axes.flatten()

for col_idx, ax_idx in zip(list(X_df_.columns), axes):
    sns.lineplot(X_df_[col_idx], ax = ax_idx)
    sns.lineplot(X_df[col_idx], ax = ax_idx)
    sns.lineplot(X_df[col_idx] - X_df_[col_idx],
c = 'k', ax = ax_idx)
    ax_idx.set_xticks([]); ax_idx.set_yticks([])
    ax_idx.axhline(y = 0, c = 'k')

# 散点图
fig, axes = plt.subplots(4,2,figsize=(4,8))
axes = axes.flatten()

for col_idx, ax_idx in zip(list(X_df_.columns), axes):
    sns.scatterplot(x = X_df_[col_idx],
y = X_df[col_idx],
ax = ax_idx)
    ax_idx.plot([-0.3, 0.3], [-0.3, 0.3], c = 'r')
    ax_idx.set_aspect('equal', adjustable='box')
    ax_idx.set_xticks([]); ax_idx.set_yticks([])
    ax_idx.set_xlim(-0.3, 0.3); ax_idx.set_ylim(-0.3, 0.3)

```

图 24. 主成分分析，使用时需要配合前文代码；

27.5 概率密度估计：高斯 KDE

本书第 12 章介绍过如何用 Seaborn 可视化高斯核密度估计结果。对于一元随机变量，高斯核密度通过在数据点附近生成高斯分布的核函数，然后将所有核函数叠加在一起得到一条曲线；这条曲线就是概率密度函数 (Probability Density Function, PDF)，用来描述样本数据的分布情况。

这一节聊一聊如何用 Statsmodels 库函数完成高斯 KDE，并可视化一元、二元概率密度函数。

一元

图 25 所示为用高斯 KDE 估计得到的鸢尾花花萼长度概率密度函数。图 25 曲线和横轴包围的面积为 1。图 26、图 27、图 28 所示为考虑鸢尾花标签的花萼长度概率密度函数。

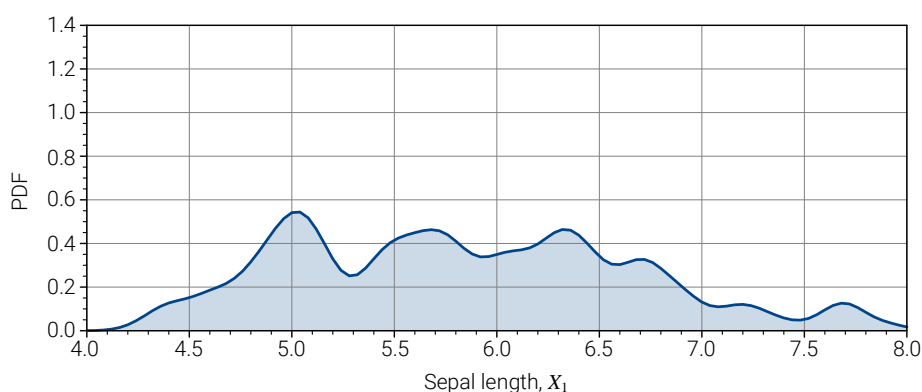


图 25. 鸢尾花数据花萼长度概率密度函数，基于高斯 KDE

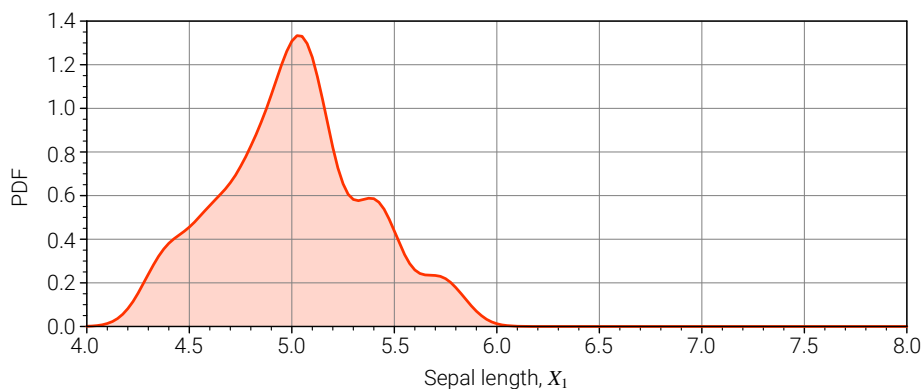


图 26. 花萼长度 X_1 概率密度函数，基于高斯 KDE，考虑标签，`'species' == 'setosa'`

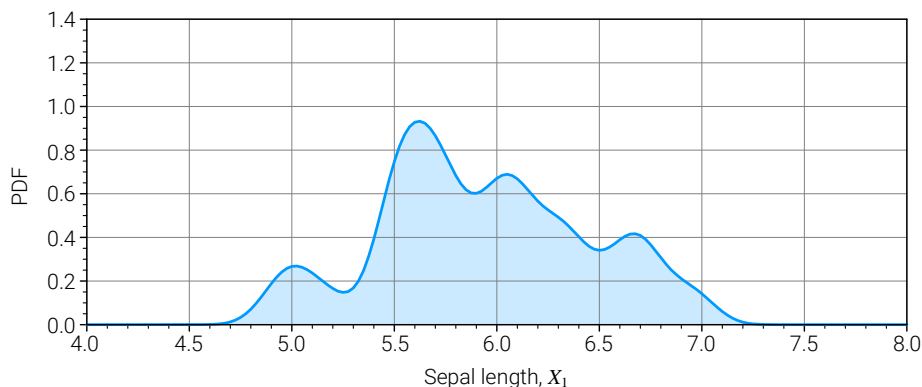


图 27. 花萼长度 X_1 概率密度函数，基于高斯 KDE，考虑标签，'species' == 'versicolor'

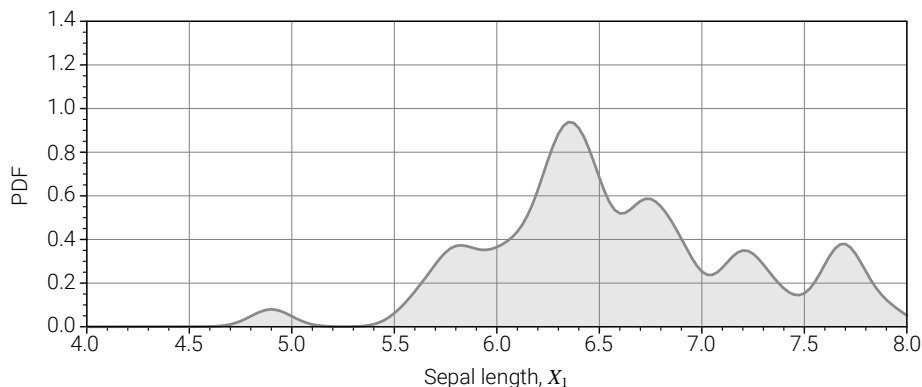


图 28. 花萼长度 X_1 概率密度函数，基于高斯 KDE，考虑标签，'species' == 'virginica'

图 29 所示为绘制图 25 ~ 图 28 的代码。下面聊一聊代码中的主要语句。

- a** 导入 statsmodels 中的 api (全称为 application programming interface) 模块。在 statsmodels 中，api 包含了用户常用的函数、类和工具，用于执行各种统计分析和建模任务。
- b** 从 sklearn.datasets 导入 load_iris。 **c** 用 load_iris() 导入鸢尾花数据集。 **d** 提取标签，这个数据集的标签为 0、1、2，分别对应 'setosa'、'versicolor'、'virginica'。 **e** 将 NumPy 数组转化为 Pandas 数据帧。 **f** 用 iloc[] 提取数据帧的第 0 列。
- g** 创建自定义可视化函数。
- h** 中 fill_between() 是 Matplotlib 库中的一个函数，用于在两条曲线之间填充颜色。
- i** 导入非参数核密度估计 sm.nonparametric.KDEUnivariate() 函数，用来创建和操作单变量数据的核密度估计对象。这个函数的输入为样本的单一变量数据。
- j** 调用 fit() 方法计算核密度估计，其中 bw 调节核函数带宽 (band width)。
- k** 利用 evaluate() 计算给定数组核密度估计值，以便后续可视化。
- l** 用自定义函数 visualize() 绘制概率密度函数曲线，'#00448A' 为一个十六进制颜色值 RGB 颜色值。在十六进制颜色表示法中，颜色值由六个字符组成，前两个字符表示红色分量、中间两个字符表示绿色分量，最后两个字符表示蓝色分量。每个字符可以取值从 00 到 FF，对应十进制的

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

0 到 255。在颜色 #00448A 中：前两个字符 00 表示红色分量为 0；中间两个字符 44 表示绿色分量为 68；最后两个字符 8A 表示蓝色分量为 138。


 创建高斯 KDE 对象时考虑鸢尾花分类。

图 30 ~ 图 33 所示为联合概率密度估计结果。



```
import numpy as np
a import statsmodels.api as sm
import matplotlib.pyplot as plt
import pandas as pd
b from sklearn.datasets import load_iris

# 从Scikit-Learn库加载鸢尾花数据
c iris = load_iris()
d y = iris.target
e X_df = pd.DataFrame(iris.data)
f X1_df = X_df.iloc[:,0]

# 自定义可视化函数
g def visualize(x1,pdf,color):
h     fig, ax = plt.subplots(figsize = (8,3))
    ax.fill_between(x1, pdf,
                    facecolor = color,alpha = 0.2)
    ax.plot(x1, pdf,color = color)

    ax.set_ylim([0,1.4])
    ax.set_xlim([4,8])
    ax.set_ylabel('PDF')
    ax.set_xlabel('Sepal length, $x_1$')

# 不考虑标签
i KDE = sm.nonparametric.KDEUnivariate(X1_df)
j KDE.fit(bw=0.1)
x1 = np.linspace(4,8,101)
k f_x1 = KDE.evaluate(x1)


l visualize(x1,f_x1,'#00448A')

# 考虑鸢尾花标签,用KDE描述样本数据花萼长度分布
colors = ['#FF3300','#0099FF','#8A8A8A']

x1 = np.linspace(4,8,161)

for idx in range(3):
m     KDE_C_i = sm.nonparametric.KDEUnivariate(X1_df[y==idx])
    KDE_C_i.fit(bw=0.1)
    f_x1_given_C_i = KDE_C_i.evaluate(x1)

    visualize(x1,f_x1_given_C_i,colors[idx])
```

图 29. 一元概率密度估计; 

二元

SciPy 也有完成概率密度估计的函数。

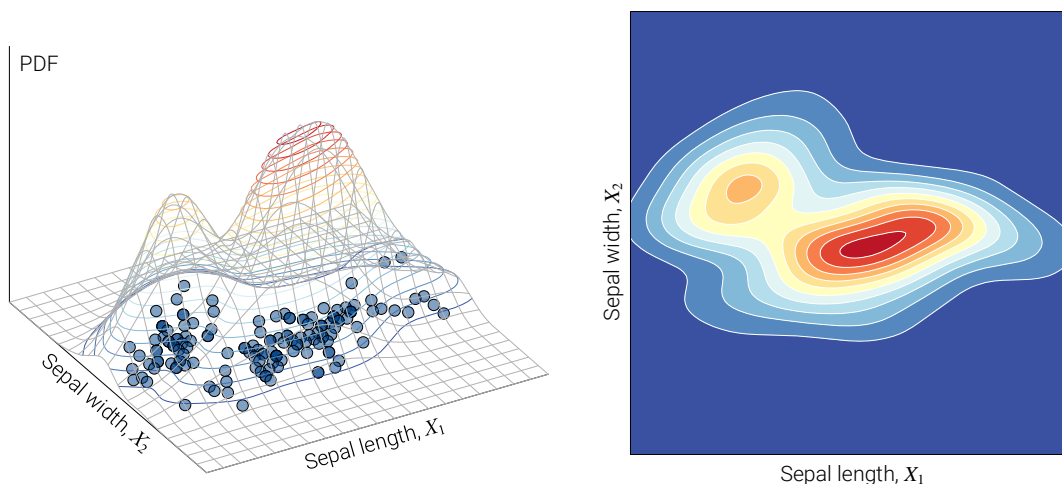
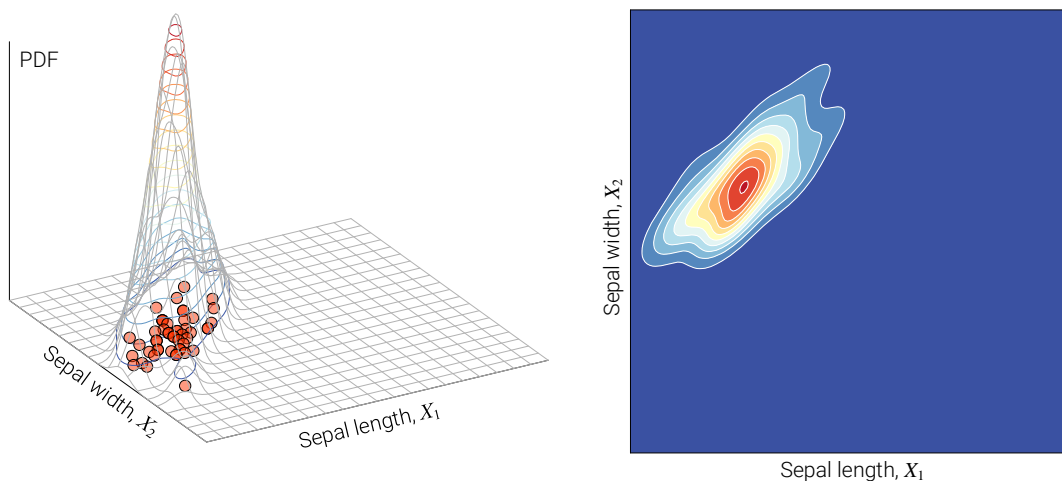
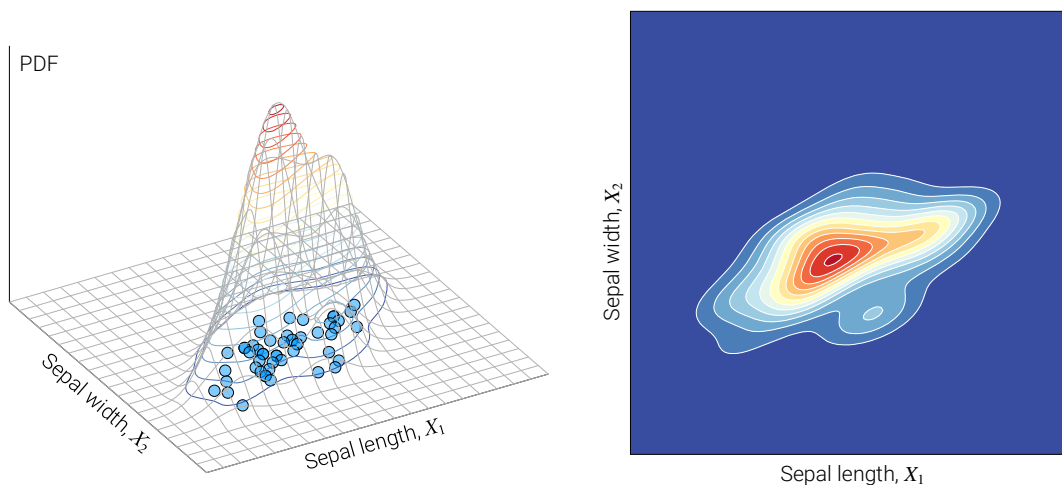
本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

图 30. 花萼长度、花萼宽度 (X_1 , X_2) 联合概率密度函数, 基于高斯 KDE图 31. 花萼长度、花萼宽度 (X_1 , X_2) 联合概率密度函数, 基于高斯 KDE, 考虑标签, 'species' == 'setosa'图 32. 花萼长度、花萼宽度 (X_1 , X_2) 联合概率密度函数, 基于高斯 KDE, 考虑标签, 'species' == 'versicolor'

本 PDF 文件为作者草稿, 发布目的为方便读者在移动终端学习, 终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有, 请勿商用, 引用请注明出处。

代码及 PDF 文件下载: <https://github.com/Visualize-ML>

本书配套微视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教, 本书专属邮箱: jiang.visualize.ml@gmail.com

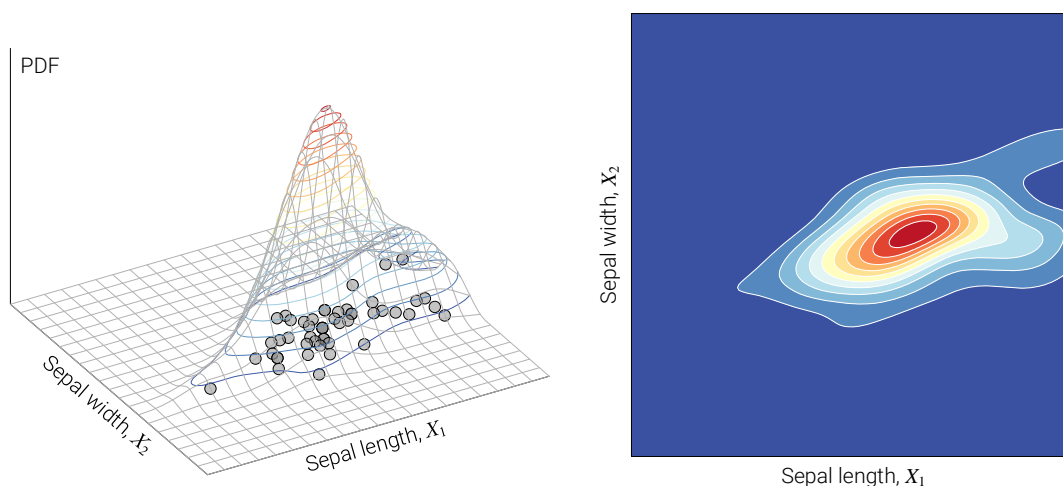


图 33. 花萼长度、花萼宽度 (X_1 , X_2) 联合概率密度函数, 基于高斯 KDE, 考虑标签, 'species' == 'virginica'

```

import matplotlib.pyplot as plt

# 定义可视化函数
a def plot_surface(xx1, xx2, surface, x1_s, x2_s,
                  z_height, color, title_txt):

b     fig = plt.figure(figsize=(8,3))


c     ax = fig.add_subplot(1, 2, 1, projection='3d')
d     ax.plot_wireframe(xx1, xx2, surface,
                       cstride = 8, rstride = 8,
                       color = [0.7,0.7,0.7],
                       linewidth = 0.25)
e     ax.scatter(x1_s, x2_s, x2_s*0, c=color)
f     ax.contour(xx1, xx2, surface, 20,
                 cmap = 'RdYlBu_r')

    ax.set_proj_type('ortho')
    ax.set_xlabel('Sepal length, $x_1$')
    ax.set_ylabel('Sepal width, $x_2$')
    ax.set_zlabel('PDF')
    ax.set_xticks([]); ax.set_yticks([])
    ax.set_zticks([])
    ax.set_xlim(x1.min(), x1.max())
    ax.set_ylim(x2.min(), x2.max())
    ax.set_zlim([0, z_height])
    ax.view_init(azim=-120, elev=30)
    ax.set_title(title_txt)
    ax.grid(False)

g     ax = fig.add_subplot(1, 2, 2)
h     ax.contourf(xx1, xx2, surface, 12, cmap='RdYlBu_r')
i     ax.contour(xx1, xx2, surface, 12, colors='w')
    ax.set_xticks([]); ax.set_yticks([])
    ax.set_xlim(x1.min(), x1.max())
    ax.set_ylim(x2.min(), x2.max())
    ax.set_xlabel('Sepal length, $x_1$')
    ax.set_ylabel('Sepal width, $x_2$')
    ax.set_aspect('equal', adjustable='box')
    ax.set_title(title_txt)

```



图 34. 定义可视化函数; 

```

import numpy as np
import statsmodels.api as sm
import pandas as pd
from sklearn.datasets import load_iris
import scipy.stats as st

# 导入鸢尾花数据
iris = load_iris()
X_1_to_4 = iris.data; y = iris.target

feature_names = ['Sepal length, $X_1$', 'Sepal width, $X_2$',
                  'Petal length, $X_3$', 'Petal width, $X_4$']

X_df = pd.DataFrame(X_1_to_4)
X1_2_df = X_df.iloc[:, [0, 1]]

x1 = np.linspace(4, 8, 161); x2 = np.linspace(1, 5, 161)
a xx1, xx2 = np.meshgrid(x1, x2)
b positions = np.vstack([xx1.ravel(), xx2.ravel()])
  colors = ['#FF3300', '#0099FF', '#8A8A8A']

c KDE = st.gaussian_kde(X1_2_df.values.T)
d f_x1_x2 = np.reshape(KDE(positions).T, xx1.shape)

x1_s = X1_2_df.iloc[:, 0]
x2_s = X1_2_df.iloc[:, 1]

z_height = 0.5
title_txt = '$f_{X1, X2}(x_1, x_2)$, evidence'
e plot_surface(xx1, xx2, f_x1_x2,
               x1_s, x2_s, z_height,
               '#00448A', title_txt)

```

图 35. 可视化证据因子，使用时配合前文代码；

```

# 考虑不同鸢尾花分类
for idx in range(3):
a   KDE_idx = st.gaussian_kde(X1_2_df[y==idx].values.T)
b   f_x1_x2_given_C_i = np.reshape(KDE_idx(positions).T, xx1.shape)

   x1_s_C_i = X1_2_df.iloc[:, 0][y==idx]
   x2_s_C_i = X1_2_df.iloc[:, 1][y==idx]

   z_height = 1
   title_txt = 'Likelihood'
c   plot_surface(xx1, xx2, f_x1_x2_given_C_i,
                 x1_s_C_i, x2_s_C_i, z_height,
                 colors[idx], title_txt)

```

图 36. 可视化似然函数，使用时配合前文代码；