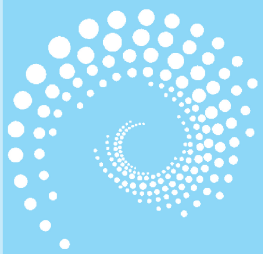
第3章

线性模型

CHAPTER 3



本章学习目标

- 认知类目标：理解回归预测的基本思想。
- 价值类目标：理解线性回归和逻辑回归的基本原理。
- 方法类目标：掌握线性回归分析进行预测方法和逻辑回归进行分类的方法。
- 情感、态度、价值观类目标：理解线性回归和逻辑回归算法在相关领域中的应用，熟练掌握线性回归和逻辑回归进行数据实际预测的案例。了解最新发展动态，能够运用相关知识分析、研究和解决实际问题，深入挖掘其在人工智能专业学科中分类或预测问题中的应用，培养学生解决实际问题的创新意识和社会责任感。

本章主要介绍回归预测的基本概念和基本思想，通过案例介绍两种常用的线性模型——线性回归和逻辑回归，分别介绍它们的基本原理和实现代码。最后介绍正则化在回归中的应用，即岭回归和套索回归。

3.1　回归预测

回归分析是数理统计中探讨两个变量或多个变量之间关系的一种常用工具,在机器学习中可以借鉴回归分析的方法对满足一定关系的变量进行建模。

最简单的回归模型是线性回归模型,即假设响应变量 Y_i 和解释变量 X_i 之间存在某种线性关系。通过建立线性模型,可以利用解释变量 X_i 来对响应变量 Y_i 进行预测。线性回归的预测值是连续的数值数据,若希望最终的预测值也是"属于某类"或者"不属于某类"这样的离散值时,可以采用逻辑回归来进行建模。本章将重点对线性回归模型和逻辑回归模型的基本原理进行解释,并通过实际案例介绍机器学习中回归分析方法的具体使用。在拓展部分将简单介绍套索回归和岭回归的基本思想和操作方法。

3.2　线性回归

3.2.1　线性回归的基本原理

线性回归的基本模型,可以用式(3.1)表示。

$$Y_i=\beta_0+\beta_1 x_{i1}+\cdots+\beta_m x_{im}+e_i,\quad i=1,2,\cdots,n \tag{3.1}$$

其中,$\hat{Y}_i=\beta_0+\beta_1 x_{i1}+\cdots+\beta_k x_{im}$ 表示模型通过解释变量 $x_{i1},x_{i2},\cdots,x_{im}$ 的计算得到的预测值;e_i 则是每个预测值 $\hat{Y}_i$ 上的随机扰动,它是一个服从正态分布 $N(0,\sigma^2)$ 的随机变量。式(3.1)中的 β_k 是回归系数,它表示每个解释变量 x_{ik} 对响应变量 Y_i 的大小,β_k 的值越大,则影响越大;β_0 是回归方程中的截距项,表示当解释变量都为零时,响应变量 $\hat{Y}_i$ 的初始值。

在回归分析的过程中,数据集中每个数据的特征都是已知的,并且在训练集上数据的特征 X_i(即解释变量)和相应的响应变量 Y_i 也是已知的,通过最小二乘法计算出式(3.1)中的回归系数 β_k 确立线性回归模型,该模型可以对响应变量未知的数据进行预测。

图 3.1 展示了采用一维线性回归模型拟合数据的实例。当训练集上的数据(x_i,y_i)点被画在直角坐标平面上,如何在拟合这些数据点的直线中找到最好的那条直线呢?这里就要确定"好"模型的标准。通过残差平方来评价回归模型拟合数据性能。

$$\frac{1}{m}\sum_{1}^{n}(Y_i-\hat{Y}_i)^2=\frac{1}{m}\sum_{1}^{n}(\hat{\beta}_0+\hat{\beta}_1 X_{1i}+\cdots+\hat{\beta}_m X_{mi})^2 \tag{3.2}$$

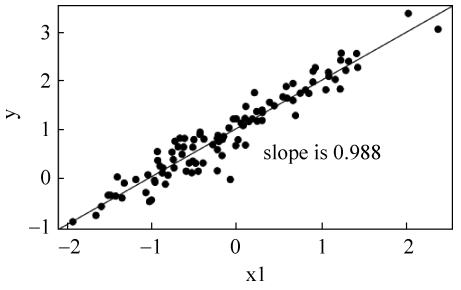


图 3.1　一维线性回归实例

式(3.2)就是线性回归的损失函数。式(3.2)取值最小的系数 $\hat{\beta}_i$ 为模型的回归系数，计算回归系数的算法通常是采用最小二乘法。残差平方和最小的本质，是要去寻找那些使响应变量的真实值和预测值差异最小的解来获得回归系数。

3.2.2　线性回归的案例——波士顿房价预测 Ⅰ

1. 案例说明

下面通过波士顿房价数据集的例子来说明线性回归模型建立的过程。

波士顿房价数据集是统计 20 世纪 70 年代中期波士顿郊区房价的中位数，数据集中包括当时郊区部分的犯罪率、房产税等共计 13 个指标(如表 3.1 所示)，以及最终的响应变量房价，试图能找到那些指标与房价之间的关系。数据集中包含 506 组数据，使用其中 480 个数据作为训练样本，剩下数据作为测试样本。

表 3.1　波士顿房价部分数据格式

列　　号	列　　名	具 体 含 义
1	CRIM	城镇人均犯罪率
2	ZN	住宅用地所占比例
3	INDUS	城镇中非商业用地所占比例
4	CHAS	查理斯河哑变量(如果边界是河流，为 1；否则，为 0)
5	NOX	一氧化氮浓度
6	RM	住宅的平均房间数
7	AGE	1940 年以前建成的自住用房比例
8	DIS	距离 5 个波士顿就业中心的加权距离
9	RAD	距离高速公路的便利指数
10	TAX	每一万美元的不动产税率
11	PTRATIO	城镇中教师学生比例
12	LSTAT	属于低收入人群房东的比例
13	MEDV	自住房的房屋均价

2. 核心代码

下面展示采用线性回归方法进行波士顿房价预测的核心代码。

(1) 导入必要的包。

```
1.  import pandas as pd
2.  from sklearn.linear_model import LinearRegression
3.  from sklearn.metrics import r2_score
4.  from sklearn.metrics import mean_squared_error
5.  import sklearn.datasets as datasets
```

(2) 获取训练数据。

```
#从 datasets 模块导入波士顿房价数据
boston = datasets.load_boston()
data = boston.data
target = boston.target
```

(3) 划分训练集和测试集,其中第 1～480 行作为训练集,第 481～506 行作为测试数据。

```
# 训练数据
X_train = data[:481]
Y_train = target[:481]
# 测试数据
X_test = data[481:]
Y_test = target[481:]
```

(4) 构建线性回归模型。

```
lr = LinearRegression()
lr.fit (X_train, Y_train)
# 查看回归模型的系数
print( "lr.coef_:{}".format(lr.coef_) )
print( "lr.intercept_:{}".format(lr.intercept_ ))
```

(5) 模型的评价。

```
lr_y_pred = lr.predict(X_test)                    #查看在测试集上模型预测的值
print(mean_squared_error(Y_test, Y_pred))         #查看模型的均方误差
print("Test set score:{:.2f}".format(lr.score(X_test, Y_test))) #查看模型在测试集上的得分
```

(6) 绘制图形。

```
import matplotlib.pyplot as plt
plt.plot(Y_test, 'o-', label = "True")
plt.plot(lr_y_pred, 'r--', label = "Line")
plt.legend()
```

3. 结果展示

模型的系数值分别为:$\beta_0 = 39.954\ 347\ 254\ 942\ 18$,$\beta_1 = -1.062\ 897\ 51e-01$,$\beta_2 = 4.326\ 563\ 33e-02$,$\beta_3 = 7.730\ 730\ 07e-04$,$\beta_4 = 2.315\ 937\ 74e+00$,$\beta_5 = -1.698\ 431\ 21e+01$,$\beta_6 = 3.307\ 384\ 65e+00$,$\beta_7 = 1.530\ 599\ 47e-02$,$\beta_8 = -1.321\ 256\ 33e+00$,$\beta_9 = -3.125\ 052\ 53e-01$,$\beta_{10} = -1.259\ 560\ 06e-02$,$\beta_{11} = -1.008\ 082\ 14e+00$,$\beta_{12} = 8.182\ 190\ 26e-03$,$\beta_{13} = -5.827\ 778\ 05e-01$。

模型的均方误差 MSE 为 19.587 266 398 327 9，模型的得分为 0.77。从这两个指标可以看出波士顿房价的线性回归模型具有较小的误差，能够较好地预测波士顿的房价数据。线性回归模型的预测值和真实值比较如图 3.2 所示。

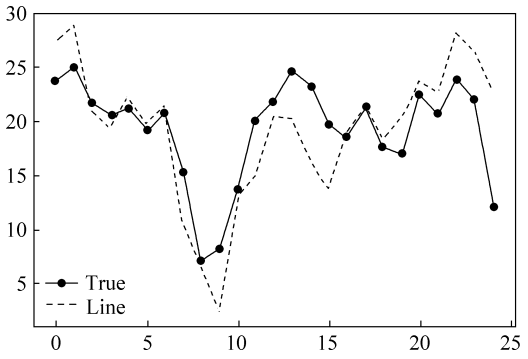


图 3.2　线性回归模型的预测值和真实值比较

3.3　逻辑回归

3.3.1　逻辑回归的基本原理

逻辑回归(Logistic Regression)是机器学习中的一种分类模型，是回归在分类问题的应用。逻辑回归主要应用于响应变量是离散值，特别是二元离散值的情况。例如，预测是否患病(患病为 1，不患病为 0)，是否为垃圾邮件(是垃圾邮件为 1，不是为 0)，是否为虚假账号等。

逻辑回归的输入数据是关于属性的特征向量。数据进来后，要经过下面三步计算。

(1) 计算线性回归的结果。

$$y=\theta_0 x_0+\theta_1 x_1+\cdots+\theta_n x_n \tag{3.3}$$

采用矩阵的形式进行描述有：

$$\boldsymbol{y}=\sum_{i=1}^{n}\boldsymbol{\theta}_i\boldsymbol{x}_i=\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x} \tag{3.4}$$

(2) 采用 Sigmoid 函数作为激活函数。

$$h_\theta(\boldsymbol{x})=g(\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x})=\frac{1}{1+\mathrm{e}^{-\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}}}$$

通过 Sigmoid 函数计算的结果是一个在[0,1]区间的概率值，如图 3.3 所示。

(3) 判断样本的最终类别。

逻辑回归最终的分类根据上一步计算出的概率值来进行预测，默认情况下阈值为 0.5。若 $h_\theta(\boldsymbol{x})>0.5$，则将样本分入正类；若 $h_\theta(\boldsymbol{x})<0.5$，则将样本分入负类。例如，有 A、B 两个类别，假设算出的概率值是输入 A 这个类别的概率值，即 A 类为正类，如果算出的概率值为 0.6，该值超过了 0.5，则样本被预测的类别为 A 类；若概率值算出来为 0.3，小于阈值 0.5，则该样本被预测为 B 类。

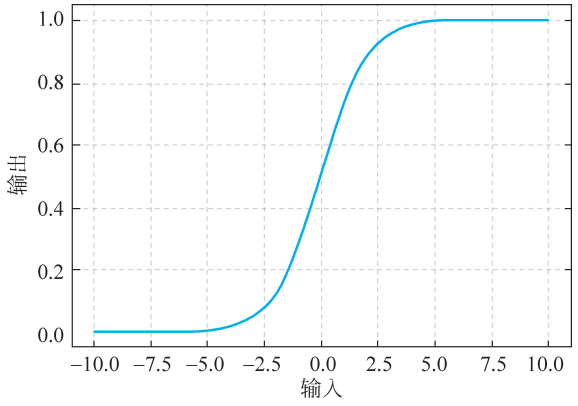


图 3.3　Sigmoid 函数图像

3.3.2　逻辑回归的损失函数

逻辑回归的损失函数为：

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}\mathrm{Cost}(h_\theta(\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)}))$$

式中，

$$\mathrm{Cost}(h_\theta(\boldsymbol{x},\boldsymbol{y}))=\begin{cases}-\log(h_\theta(\boldsymbol{x})), & \text{如果 } y=1\\ -\log(1-h_\theta(\boldsymbol{x})), & \text{如果 } y=0\end{cases}$$

这样构建的 $\mathrm{Cost}(h_\theta(\boldsymbol{x},\boldsymbol{y}))$函数的特点如下。

(1) 当实际 $y=1$ 且 h_θ 也为 1 时，损失为 0；当 $y=1$ 但 h_θ 不为 0 时，损失会随着 h_θ 的变大而变大，即真实值与预测值差距变大。

(2) 当实际 $y=0$ 且 h_θ 也为 0 时，损失为 0；当 $y=0$ 当 h_θ 不为 0 时，损失会随着 h_θ 的变大而变大，即真实值与预测值差距变大。

可将构建的 $\mathrm{Cost}(h_\theta(\boldsymbol{x},\boldsymbol{y}))$函数等价地写成如下形式：

$$\mathrm{Cost}(h_\theta(\boldsymbol{x},\boldsymbol{y}))=-y\times\log(h_\theta(\boldsymbol{x}))-(1-y)\times\log(1-h_\theta(\boldsymbol{x}))$$

代入上面公式中，得到逻辑回归的损失函数 $J(\theta)$的表达式为：

$$J(\theta)=-\frac{1}{m}\left[\sum_{i=1}^{m}y^{(i)}\log h_\theta(x^{(i)})+(1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

3.3.3　逻辑回归的案例——泰坦尼克号乘客预测

1. 案例说明

泰坦尼克号乘客数据集是数据科学竞赛平台 Kaggle 上最经典的数据集之一。这个数据集中响应变量 Survived 的取值为是否生还，1 为生还，0 为未生还。在本案例中要根据乘客数据的信息，来预测其是否能够生还。因此该问题是一个经典的分类问题，可以采用逻辑回归来进行建模和预测。

泰坦尼克号乘客数据集的下载网址是 https://www.kaggle.com/c/titanic。数据集中的具体字段如表 3.2 所示。

表 3.2　泰坦尼克号乘客数据集格式

列　　号	列　　名	具 体 含 义
1	PassengerId	乘客编号
2	Survived	是否生还(1 为生还,0 为未生还)
3	Pclass	船舱等级,类别数据
4	Name	乘客姓名
5	Sex	乘客性别,类别数据
6	Age	乘客年龄,数值数据
7	SibSp	乘客在船上的兄弟姐妹及配偶数量,数值数据
8	Parch	乘客在船上的父母和子女数量,数值数据
9	Ticket	船票编号
10	Fare	票价
11	Cabin	舱位
12	Embarked	登船港口,类别数据

2. 核心代码

下面展示采用逻辑回归方法进行泰坦尼克号乘客生存与否预测的核心代码。

(1) 导入必要的包。

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

(2) 导入数据。

```
df = pd.read_csv('.\data\Titanic.csv')
```

(3) 查看数据的基本信息。

```
df.info()
df.head()
```

查看数据的情况如图 3.4 和图 3.5 所示。

从 Titanic.csv 的数据集上可以看到 Age 字段有 177 个空缺值,Sex 字段的值有两个,分别是 male 和 female,要将这两个值转换为数值型数据。

(4) 完成数据预处理。

```
#对 Sex 字段进行编码,male 是 0,female 是 1
df['Sex'] = df['Sex'].map({'male':0, 'female':1})
# 用均值填充 Age 字段的空缺值
df.Age.fillna( value = df.Age.mean(), inplace = True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

图 3.4　查看数据的基本信息

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

图 3.5　查看数据的前 5 条记录

(5) 划分训练集和测试集。

```
# 选择 Pclass,Sex,Age,SibSp,Parch 和 Fare 作为自变量,Survived 作为预测变量
titanic = df[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Survived']]
X = titanic.iloc[:,0:6]
y = titanic.iloc[:,6:7]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

(6) 特征矩阵进行归一化处理。

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

(7) 建立逻辑回归模型。

```
from sklearn.linear_model import LogisticRegression
```

```
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
# 获得测试集上的预测值
y_pred = classifier.predict(X_test)
```

（8）评价模型的性能。

```
# 计算模型的分类正确率
from sklearn.metrics import accuracy_score
printf( accuracy_score(y_test, y_pred) )      # 在测试集的预测准确率约为 79 %
# 计算模型的混淆矩阵
from sklearn.metrics import confusion_matrix
df = pd.DataFrame(
    confusion_matrix(y_test, y_pred),
    columns = ['预测未幸存', '预测幸存'],
    index = ['实际未幸存', '实际幸存']
)
printf( df )
# 绘制 ROC 曲线,计算 ROC 曲线下面积
from sklearn.metrics import plot_roc_curve, roc_curve, auc, roc_auc_score
# 绘制 ROC 曲线
fig,ax = plt.subplots(figsize = (6,4))
lr_roc = plot_roc_curve( estimator = classifier, X = X_test, y = y_test, ax = ax,
linewidth = 1)
# 计算 ROC 曲线下面积
printf( roc_auc_score(y_test,y_pred))        # 0.775
plt.show()
```

3. 结果展示

结果如图 3.6 和图 3.7 所示。

	预测未幸存	预测幸存
实际未幸存	94	16
实际幸存	21	48

图 3.6　模型的混淆矩阵

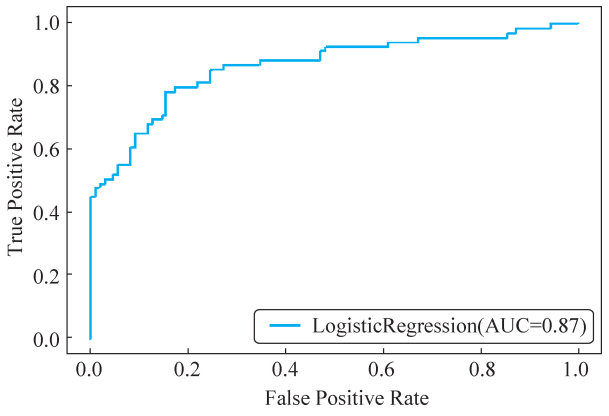


图 3.7　模型的 ROC 曲线

3.4 其他回归模型

当用线性回归解决某些机器学习问题时,会遇到过拟合问题。造成过拟合的原因有很多,比如数据的特征之间存在较强的多重共线性,数据特征较多而训练数据较少,或者数据存在较多噪声等情况。当训练数据训练出的模型参数多导致模型复杂,模型在训练集上拟合效果较好,但是在测试集上预测效果很差时,往往出现了过拟合。

解决过拟合的方法有很多种,正则化是处理过拟合问题的重要手段之一。在大部分机器学习模型的损失函数中,可以通过添加正则项来缩小参数值。在线性回归模型的损失函数中,若添加 L1 正则项,则称为套索(Lasso)回归;若添加一个 L2 正则化,则称为岭(Ridge)回归。

3.4.1 套索回归

套索回归即是在线性回归模型的损失函数中添加 L1 范数。套索回归的损失函数为:

$$\min_{\omega} \frac{1}{n} \| X\omega - Y \|_2^2 + \alpha \| \omega \|_1$$

L1 正则化可以产生稀疏的模型,使得一些特征项的系数为 0,因而可以用于选择特征,只留下系数不为 0 的特征。

3.4.2 岭回归

岭回归是在损失函数中添加 L2 范数的平方作为正则项来训练模型。岭回归的损失函数为:

$$\min_{\omega} \frac{1}{n} \| X\omega - Y \|_2^2 + \alpha \| \omega \|_2^2$$

L1 正则化会趋向于选择较少的特征,从而使得其他特征的系数都为 0,而 L2 正则化会选择更多特征,从而这些特征的系数都会接近 0。模型拟合过程中通常倾向于让参数尽可能小,数据即使发生了一定的偏移也不会对拟合结果造成太大影响,从而提升模型的泛化性能。

3.4.3 套索回归和岭回归的案例——波士顿房价预测 Ⅱ

还是使用波士顿房价案例来解释岭回归和套索回归编码过程。

(1) 导入必要的包。

```
1.  from sklearn.datasets import load_boston
2.  from sklearn.preprocessing import MinMaxScaler, PolynomialFeatures
3.  from sklearn.model_selection import train_test_split
4.  import numpy as np
```

（2）加载波士顿房价数据集。

```
def load_extended_boston():
        boston = load_boston()
X = boston.data
# 数据归一化处理
        X = MinMaxScaler().fit_transform(boston.data)
        # 波士顿房价数据集的特征增加到了 105 个
        X = PolynomialFeatures(degree=2, include_bias=False).fit_transform(X)
return X, boston.target

X, y = load_extended_boston()
```

（3）划分训练数据和测试数据。

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

（4）构建模型。

```
# 构建线性回归模型
from sklearn.linear_model import LinearRegression
lr = LinearRegression().fit(X_train, y_train)
print("Training set score:{:.2f}".format(lr.score(X_train, y_train)))
                                                    # 训练集的测试分数
print("Test set score:{:.2f}".format(lr.score(X_test, y_test)))        # 测试集的测试
# Training set score:0.95,Test set score:0.61,说明出现过拟合

    # 构建岭回归模型
from sklearn.linear_model import Ridge
ridge = Ridge().fit(X_train, y_train)                  #正则化系数 alpha,默认为 1.0
print("Training set score:{:.2f}".format(ridge.score(X_train, y_train)))
                                                    # 训练集的测试分数
print("Test set score:{:.2f}".format(ridge.score(X_test, y_test)))
                                                    # 测试集的测试分数
# Training set score:0.89,Test set score:0.75

ridge = Ridge(alpha=0.8).fit(X_train, y_train)
print("Training set score:{:.2f}".format(ridge.score(X_train, y_train)))
                                                    # 训练集的测试分数
print("Test set score:{:.2f}".format(ridge.score(X_test, y_test)))
                                                    # 测试集的测试分数
# 减小 alpha 可以让系数(w)受到的限制更小
# 当 alpha 特别小的时候就与普通的 LinearRegression 没什么区别

# 构建套索回归模型
from sklearn.linear_model import Lasso
lasso = Lasso().fit(X_train, y_train)                  #正则化系数 alpha,默认为 1.0
print("Training set score:{:.2f}".format(lasso.score(X_train, y_train)))
print("Test set score:{:.2f}".format(lasso.score(X_test, y_test)))
```

```
print("Number of features used:{}".format(np.sum(lasso.coef_ != 0)))
 # Training set score:0.29,Test set score:0.21,
 # Number of features used:4
 # alpha = 1.0 的套索回归模型得分很低,此时模型的特征缩减到 4 个

lasso001 = Lasso(alpha = 0.01, max_iter = 100000).fit(X_train, y_train)
print("Training set score:{:.2f}".format(lasso001.score(X_train, y_train)))
print("Test set score:{:.2f}".format(lasso001.score(X_test, y_test)))
print("Number of features used:{}".format(np.sum(lasso001.coef_ != 0)))
 # Training set score:0.90,Test set score:0.77
 # Number of features used:33
 # alpha = 0.01 的套索回归模型得分为 0.77,,此时模型的特征有 33 个
```

3.5　知识扩展

目前,线性模型正则化主要用 L1、L2 正则化。除此之外,还存在另外的正则化技术,需要视具体问题而定,可以查阅相关正则化方面的文献。

3.6　习题

1. 简述线性回归的基本原理。
2. 简述逻辑回归的基本原理。
3. 对鲍鱼数据集分别采用线性回归、岭回归和套索回归来进行建模,并比较模型性能。