案例七：预测皮马印第安人是否有糖尿病

1. 案例目的

• 理解逻辑回归模型的原理；

• 理解递归特征消除算法的原理。

2. 案例内容

基于逻辑回归模型进行预测皮马印第安人是否有糖尿病。

3. 案例知识点

• 逻辑回归算法基本原理；

• 递归特征消除算法基本原理。

4. 案例时长

共1学时，具体安排如下：

• 数据预处理（0.3学时）

• 对数据进行降维（0.5学时）

• 构建模型并评估（0.2学时）

5. 案例实验环境

**•操作系统：**

1）Linux Ubuntu 16.04 64位操作系统

**•软件环境：**

1）Python 3.7.4

2）sklearn 0.23.1

**•开发环境与工具：**

1）Spyder 4

6. 案例分析

本案例主要分为以下3部分：

1）数据预处理，为构建模型做准备；

2）对数据进行降维，提升模型性能；

3）构建模型并评估性能；

7. 案例实验过程

基于逻辑回归预测皮马印第安人是否有糖尿病，可分为3个大的步骤：

1、数据预处理；

1.1 导入包

1.2 导入数据集

1.3 处理缺失数据

1.4 将数据分成自变量和因变量

1.5 拆分数据集

1.6 特征缩放

2、对数据进行降维；

2.1 使用递归特征消除算法进行降维

2.2 可视化字段个数 VS 准确率分数

2.3 了解降维后的信息并得到新的自变量

3、构建模型并评估性能；

3.1 构建逻辑回归模型

3.2 预测测试集

3.3 模型分类报告

7.1开发准备

在桌面上新建一个目录，名字叫pima-indians-diabetes。本案例中所有的数据集和代码都放在该目录下。

7.1.1 打开Spyder软件

在桌面上点击Spyder图标打开Spyder，并把工作目录切换至pima-indians-diabetes目录下。

7.1.2 获取数据

本案例需要1个数据集，pima-indians-diabetes.csv。

点击右侧菜单栏中的【资源下载】按钮，复制提供的URL前缀，如https://xxx.eec-cn.com/。

将URL前缀和数据集地址拼接到一起，如https://xxx.eec-cn.com/dataSet/systemLib/15a4d328e7bf47bfacd1426a3ee0ff7c.csv。下载数据集，将pima-indians-diabetes.csv复制到刚刚创建的pima-indians-diabetes目录下。

7.1.3 新建代码文件

确保Spyder的工作目录为pima-indians-diabetes。使用Spyder新建1个Python文件，命名为pima-indians-diabetes.py。至此，整个案例的目录结构图1所示。

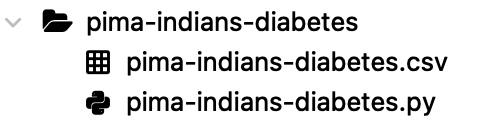


图1 案例的目录结构

7.1.4 了解数据

pima-indians-diabetes.csv数据集包含768条数据，9个字段。这9个字段的含义分别为：

• preg：数值类型；怀孕次数

• plas：数值类型；口服葡萄糖耐量测试中2小时的血浆葡萄糖浓度

• pres：数值类型；血压

• skin：数值类型；皮肤厚度

• test：数值类型；胰岛素

• mass：数值类型；体重指数

• pedi：数值类型；糖尿病谱系功能

• age：数值类型；年龄

• class：字符串类型；测试结果；有2个值，1：有糖尿病；0：没有糖尿病

7.2数据预处理

向pima-indians-diabetes.py添加代码。

7.2.1 导入包

【代码7-2-1】pima-indians-diabetes.py

# 导入包

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.feature\_selection import RFECV

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report

执行代码块。

7.2.2 导入数据集

【代码7-2-2】pima-indians-diabetes.py

# 导入数据集

dataset = pd.read\_csv('pima-indians-diabetes.csv')

执行代码块。

从现在开始，建议打开Spyder的Variable explorer面板，实时查看变量的信息。

dataset变量的信息如图2所示。

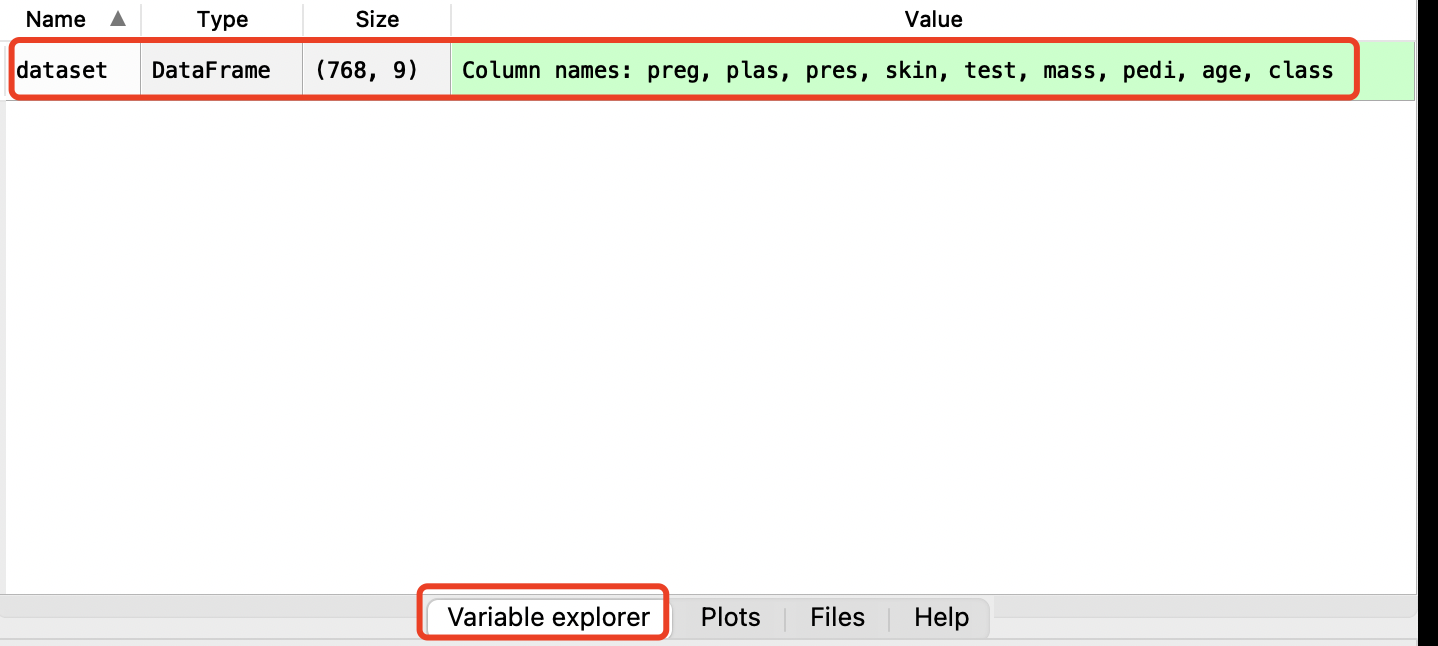


图2 dataset变量的信息

由上图可见，数据集有768行，9列。

7.2.3 处理缺失数据

统计缺失数据。

【代码7-2-3】pima-indians-diabetes.py

# 处理缺失数据

null\_df = dataset.isnull().sum()

执行代码块。

通过查看null\_stat变量的值发现，没有缺失数据，如图3所示。

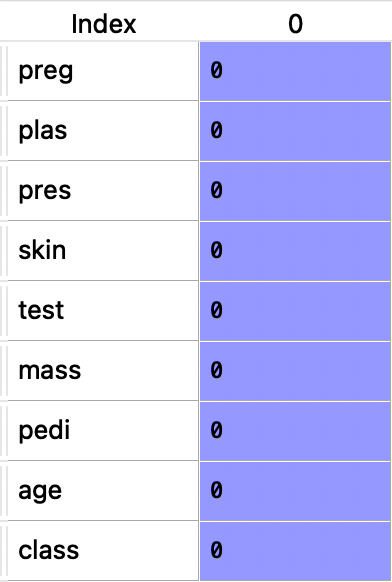


图3 缺失数据统计结果

7.2.4 将数据分成自变量和因变量

现在数据集有9个列，将class列作为因变量，其余的8个列作为自变量。

【代码7-2-4】pima-indians-diabetes.py

# 将数据分成自变量和因变量

x = dataset.iloc[:,0:8].values

y = dataset.iloc[:,8].values

执行代码块。

7.2.5 拆分数据集

将数据集拆分成训练集和测试集，比例为80%，20%。

【代码7-2-5】pima-indians-diabetes.py

# 拆分数据集

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.2, random\_state = 1)

执行代码块。

7.2.6 特征缩放

将大部分数据都控制在-3至3之间，这将加快模型训练速度。

【代码7-2-6】pima-indians-diabetes.py

# 特征缩放

sc\_x = StandardScaler()

x\_train = sc\_x.fit\_transform(x\_train)

x\_test = sc\_x.transform(x\_test)

执行代码块。

7.3对数据进行降维

7.3.1 使用递归特征消除算法进行降维

由于数据过多，并且不是每一个字段都有用，所以我们对数据进行降维，提升之后构建模型的性能。

【代码7-3-1】pima-indians-diabetes.py

# 使用递归特征消除算法进行降维

model = LogisticRegression(random\_state=1)

rfecv = RFECV(estimator = model, min\_features\_to\_select = 1, cv=5, verbose=1, step=1, scoring='accuracy')

rfecv = rfecv.fit(x\_train, y\_train)

执行代码块。

7.3.2 可视化字段个数 VS 准确率分数

【代码7-3-2】pima-indians-diabetes.py

# 可视化字段个数 VS 准确率分数

plt.figure()

plt.xlabel("Number of Features")

plt.ylabel("Accuracy")

plt.plot(range(1, len(rfecv.grid\_scores\_) + 1), rfecv.grid\_scores\_)

plt.show()

执行代码块。

执行结果如图4所示。

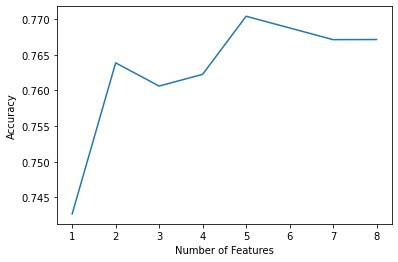


图4 字段个数 VS 准确率分数

由上图可见，在字段个数为5时，准确率分数最高。因此我们筛选出这5个字段进行构建模型。

7.3.3 了解降维后的信息并得到新的自变量

【代码7-3-3-1】pima-indians-diabetes.py

# 打印其它降维后的信息

print("选择的字段个数：%d" % rfecv.n\_features\_) # 5

print("选择的字段索引是：%s" % rfecv.support\_) # [ True True True False False True True False]

print("字段排名是：%s" % rfecv.ranking\_) # [1 1 1 4 3 1 1 2]

执行代码块。

我们可以看到，选择了5个字段。

接下来我们筛选出这5个字段作为新的自变量。

【代码7-3-3-2】pima-indians-diabetes.py

# 得到降维后的自变量

x\_train\_selected = x\_train[:, [0,1,2,5,6]]

x\_test\_selected = x\_test[:, [0,1,2,5,6]]

执行代码块。

7.4构建模型并评估性能

7.4.1 构建逻辑回归模型

【代码7-4-1】pima-indians-diabetes.py

# 构建逻辑回归模型

model = LogisticRegression(random\_state=1)

model.fit(x\_train\_selected, y\_train)

执行代码块。

7.4.2 预测测试集

【代码7-4-2】pima-indians-diabetes.py

# 预测测试集

y\_predict = model.predict(x\_test\_selected)

执行代码块。

预测结果如图5所示。

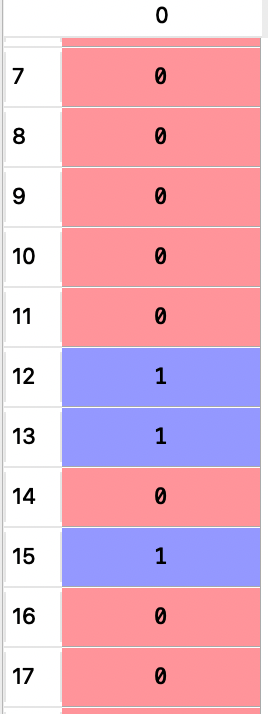


图5 预测结果

7.4.3 模型分类报告

【代码7-4-3】pima-indians-diabetes.py

# 模型分类报告

print(classification\_report(y\_test, y\_predict))

执行代码块。

模型分类报告如图6所示。

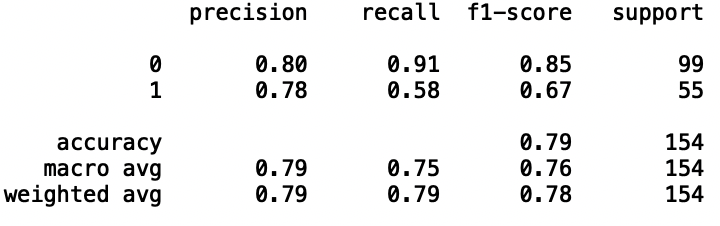


图6 模型分类报告

由上图可见，f1-score指标的分数为0.78，说明模型性能还不错。

8. 案例代码

【案例代码】pima-indians-diabetes.py

# 导入包

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.feature\_selection import RFECV

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report

# 导入数据集

dataset = pd.read\_csv('pima-indians-diabetes.csv')

# 处理缺失数据

null\_df = dataset.isnull().sum()

# 将数据分成自变量和因变量

x = dataset.iloc[:,0:8].values

y = dataset.iloc[:,8].values

# 拆分数据集

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.2, random\_state = 1)

# 特征缩放

sc\_x = StandardScaler()

x\_train = sc\_x.fit\_transform(x\_train)

x\_test = sc\_x.transform(x\_test)

# 使用递归特征消除算法进行降维

model = LogisticRegression(random\_state=1)

rfecv = RFECV(estimator = model, min\_features\_to\_select = 1, cv=5, verbose=1, step=1, scoring='accuracy')

rfecv = rfecv.fit(x\_train, y\_train)

# 可视化字段个数 VS 准确率分数

plt.figure()

plt.xlabel("Number of Features")

plt.ylabel("Accuracy")

plt.plot(range(1, len(rfecv.grid\_scores\_) + 1), rfecv.grid\_scores\_)

plt.show()

# 了解降维后的信息并得到新的自变量

# 打印其它降维后的信息

print("选择的字段个数：%d" % rfecv.n\_features\_) # 5

print("选择的字段索引是：%s" % rfecv.support\_) # [ True True True False False True True False]

print("字段排名是：%s" % rfecv.ranking\_) # [1 1 1 4 3 1 1 2]

# 得到降维后的自变量

x\_train\_selected = x\_train[:, [0,1,2,5,6]]

x\_test\_selected = x\_test[:, [0,1,2,5,6]]

# 构建逻辑回归模型

model = LogisticRegression(random\_state=1)

model.fit(x\_train\_selected, y\_train)

# 预测测试集

y\_predict = model.predict(x\_test\_selected)

# 模型分类报告

print(classification\_report(y\_test, y\_predict))