第十二章 数据预测建模实验

上一章讨论了数据的探查建模，也就是给一个数据集合，没有任何其它可以了解的知识，仅仅是希望探查这些数据有什么规律性的东西，也就是这些数据有什么样的结构组织形式，比如使用聚类分析就发现 wine 数据集中的所有样本根据其酒中的成分，可以聚为三类，每一类酒具有相似的成分。这就说明我们成功地从这批数据中探查到了一个模型，那就是这些不同类型的酒其实可以分为三类，如果要评估酒的营养成分，只要从三类中各抽取一个样本就可以了。

现在给出一个刚刚研发出来的新酒，判断它到底属于上面三类中的哪一类酒，这时候怎么办呢？我们立刻想到可以计算这个新酒和上面三类酒中心点的距离，距离那个中心点近就属于那类酒，这就是预测了，也就是给定一个样本，能够立刻推断出其所属类别。

信息处理的末端就是决策，决策的结果就是给出一个是和非的判断。我们在具体生产活动中得出大量样本数据，也知道它的性质，希望从这些数据中建立一个模型，以此可以预测给定新样本的性质。比如一个医院，获得了很多关于肿瘤的数据，判断肿瘤的指标有很多个，比如肿块厚度，细胞大小，细胞形状的均匀性，边际附着力，单个上皮细胞大小等等，根据这些指标获得一批样本数据，一些样本最后发现是恶性肿瘤，一些样本最后发现是良性肿瘤。这就意味着已经获得了关于这批数据的分类特征，现在来了一个病人，化验得出关于肿瘤的多个指标数据，那么如何判断这个病人的肿瘤是良性肿瘤还是恶性肿瘤呢？这就是预测了，也就是根据已有的样本所属类别，预测未知样本的所属类别。这种判断决策分类预测有点特殊，就是所有的样本只分为两类，对应实际含义就是“是”和“非”。现实生活中这类例子太多了：暑假来了，根据过去销售经验，哪类学生才会买笔记本电脑；银行放贷，根据用户过去的信用和资产等情况，哪些用户是潜在客户；邮件过滤，根据关键词，图像和超文本等数据，哪些邮件是垃圾邮件等等，这些都要做出判断，都是根据过去的知识，对给定的观测值做出“是”或者“非”的判断，进而进行决策。

对已知类别的样本建模，已经提出了很多算法，比如线性判别分类，KNN、SVM、朴素贝叶斯分类、人工神经网络、决策树、C5.0、随机森林、adaboost 和 bagging 等等许多种算法。这些算法构建分类模型的步骤一般都是先将已知样本数据分为两批，比如随机挑选三分二的样本作为训练样本进行建模，然后使用剩下的三分一的样本代入模型进行预测，看预测的分类结果和已知的分类是否一致，这就是分类验证，如果大多数都正确的分类了，那么这个模型就证明是有效的，可以使用了，如果大多数分错了，那么这个模型是无效的，可能要使用其它模型来建模了。本章实验的编写主要参考了文献[20]和文献[23]等。

第一节 神经网络和 Keras 模型

一、神经网络的基本原理

1. 神经网络的结构

人工神经网络（Artificial Neural Network, ANN）是受生物神经系统启发而建立的数学模型。它由大量的神经元（Neuron）组成，通常分为三层，即

输入层（Input Layer）：接收原始数据，例如样本的特征。一般节点数目和输入数据的个数相同，只能是数值，其它数据需要进行数值转换；

隐藏层（Hidden Layer）：对输入进行非线性变换和特征提取。可以有一层或多层。每一层的节点数目根据实际情况选取；

输出层（Output Layer）：给出预测结果，例如分类概率或回归值。如果是分类，则一般计算输出概率值，如果是回归，则给出输出原始数据。根据输出数据计算耗费函数。

图 12.1 就是一个三层神经网络图示，一个输入层，一个隐藏层，一个输出层，每一层的节点数目可以选择若干个，层与层之间的节点可以全连接也可以部分链接。节点的数目以及隐藏层的层数需要根据实际情况进行评估选择，目前还没有一个确定的依据。

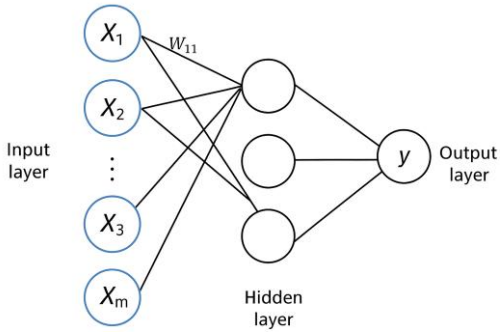


图 12.1 三层神经网络图

一个简单的前馈神经网络结构如下：输入层 → 隐藏层 → 输出层。每个神经元之间通过权重（Weights）和偏置（Bias）连接，神经元的计算公式为：

$$y_i = \sum_{i=1}^{n} w_{ij}x_j + b_i$$

然后通过激活函数（Activation Function）进行非线性变换

$$z_i = f(y_i)$$

常见的激活函数有：

Sigmoid：输出范围 (0,1)，常用于二分类。

ReLU：线性整流函数，常用于隐藏层。

Softmax：将输出转换为概率分布，常用于多分类。

2、神经网络的训练过程

神经网络的训练本质上是一个参数优化问题，主要步骤如下：

前向传播（Forward Propagation）：从输入层到输出层逐层计算，得到预测值。

损失函数（Loss Function）：计算预测值与真实值之间的误差。例如：二分类：交叉熵损失（Binary Crossentropy）多分类：交叉熵损失（Categorical Crossentropy）；回归：均方误差（MSE）

反向传播（Backpropagation）：根据误差计算梯度。参数更新：通过优化算法（如梯度下降、Adam）更新权重和偏置，使损失逐渐减小。

训练过程通常会迭代多次（称为 epoch），直到模型收敛或达到设定的训练次数。

二、Keras 实现神经网络的基本函数和用法

Keras 是一个高层神经网络 API，能够快速搭建和训练深度学习模型。它的底层可以调用 TensorFlow。

Keras 常用模块：Sequential：顺序模型，层按顺序叠加；Dense：全连接层（最常见的层类型）；Activation：激活函数；compile()：配置模型，定义损失函数、优化器、评价指标；fit()：训练模型；evaluate()：评估模型；predict()：预测新数据。

三、神经网络实现模型实验

本文选择一个乳腺癌数据集，共有 699 个样本，每一个样本有 9 个指标，一个表示良性还是恶性的分类号，具体列名称为：ID,clumpThickness,sizeUniformity, shapeUniformity, maginalAdhesion,

singleEpithelialCellSize, bareNuclei, blandChromatin, normalNucleoli, mitosis, class。做实验的时候需要将第一列 ID 号，最后一列类号去掉，只使用中间的数据进行计算。

使用这些数据进行神经网络建立分类模型的时候，可以选其中 70%为训练样本来建立预测模型，30%的样本来验证模型是否有效，当然这种选择应该是随机的。具体实现代码如下

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix,accuracy_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2' # 屏蔽警告信息
import readDataFromWugangTOP as wt
url = "https://wugang69.top/dataset/breast.csv"
breast = wt.read_csv_with_headers(url)

breast.replace('NA', np.nan, inplace=True)
# 删除缺失值
breast.dropna(inplace=True)
# 将 bare_nuclei 列转换为整数
breast['bare_nuclei'] = breast['bare_nuclei'].astype(int)
# 分离特征和标签
X = breast.iloc[:, 1:10].values  # 去掉 id 列
y = breast['class'].values       # 类别 2=良性，4=恶性
# 将 class 转换为 0/1 (2 -> 0, 4 -> 1)
y = np.where(y == 2, 0, 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = Sequential([Dense(16, input_dim=X_train.shape[1], activation='relu'),
                    Dense(8, activation='relu'),
                    Dense(1, activation='sigmoid')
                    ])
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=50, batch_size=16, verbose=1, validation_split=0.1)

# 预测概率
y_pred_prob = model.predict(X_test)
# 转换为 0/1
y_pred = (y_pred_prob > 0.5).astype(int).flatten()

# 混淆矩阵
cm = confusion_matrix(y_test, y_pred)
print("混淆矩阵:")
print(cm)

acc = accuracy_score(y_test,y_pred)
print("神经网络(16,8) 聚类准确率: {:.2f}%".format(acc*100))
```

上面的神经网络模型选择两个隐藏层，第一个隐藏层设置为 16 个节点，第二个隐藏藏设置为 8 个节点。隐藏层的激活函数选择为 relu 函数，输出层的激活函数为 sigmoid。耗费函数选择为 binary_crossentropy 函数。在神经网络训练迭代训练的时候，设置为 50 个周期，每次输入 16 个样本一起运算，权值梯度优化参数方法选择为 adam 算法。经过 50 轮训练，建模神经网络模型，然后对测试样本进行预测，具体分类的混淆矩阵为

```
[[125   2]
 [  6  72]]
```

可以观察到，有 2 个恶性肿瘤被误分类到良性，有 6 个良性被误分类到恶性中。可以计算分类准确率，就是(125+72)/205=96%。需要说明的是验证样本实际上为 210，而现在使用的验证样本为 205，这是因为其中的 5 个样本由于缺项没有参加验证。这个模型 96%的有效性还是可以接受的。可以增减隐藏层以及隐藏层节点数目，以及其他防止过拟合等算法来优化神经网络模型。

第二节 决策树

本节介绍经典决策树，主要用于根据预测变量预测多元因变量，这是经典决策树特点，其具体算法步骤就是(1)可以根据已知训练样本的分类情况，计算所有预测变量的信息增益，将具有信息增益最大的预测变量作为测试属性，创建一个节点，将所有样本按照该预测变量取值进行分组；(2)每一组样本数据继续按照(1)的步骤进行；(3)重复上面两个步骤；(4)结束的条件就是给定节点的所有样本都属于同一个类，或者没有预测变量了，或者分支没有样本。

这种决策树建模算法可能造成过拟合现象，过拟合现象出现的原因就是建模时将噪声和误差都当作真实信息参与分类，因此建立的模型用来预测样本数据，其效果会很差。为了解决这个问题，通常需要对生成的决策树进行剪枝。

我们还是以上一节的肿瘤为例，使用决策树对该肿瘤数据建立预测模型，即使用上一节训练样本构造决策树模型。Python 语言中包中的 sklearn.tree.DecisionTreeClassifier()函数可以实现决策树构建，prune()函数可以实现对构建的决策树进行剪枝，从而得到更优的决策树。该函数一般参数形式为：

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(
    criterion="gini",
    splitter="best",
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    min_weight_fraction_leaf=0.0,
    max_features=None,
    random_state=None,
    max_leaf_nodes=None,
    min_impurity_decrease=0.0,
    class_weight=None,
    ccp_alpha=0.0
)
```

1. criterion （默认 "gini"）：指定分裂节点时的“划分标准”。可选值："gini"：基尼不纯度（Gini Impurity）。"entropy"：信息增益（Information Gain）。"log_loss"：基于对数损失（类似于交叉熵）。

2. splitter （默认 "best"）：指定每个节点划分的策略。可选值："best"：在所有特征中选择最优划分点。"random"：随机选择部分特征中的划分点（增加多样性）。

3. max_depth （默认 None）：树的最大深度。限制深度可以防止过拟合。若为 None，则一直分裂直到叶子节点纯净，或者样本数小于 min_samples_split。

4. min_samples_split （默认 2）：一个节点要继续划分所需的最小样本数。可以是：整数（例如 10，表示至少 10 个样本才分裂）。浮点数（例如 0.1，表示至少占总样本数的 10% 才分裂）。

5. min_samples_leaf （默认 1）：每个叶子节点所需的最少样本数。可以是整数或浮点数。调大 min_samples_leaf 可以让树更平滑，减少过拟合。

6. min_weight_fraction_leaf （默认 0.0）：与 min_samples_leaf 类似，但要求叶子节点的 样本权重占比 不小于该值。一般在样本有权重时使用。

7. max_features （默认 None）：划分节点时考虑的最大特征数。可选值：None：使用所有特征。"sqrt"：使用特征数的平方根（常用于随机森林）。"log2"：使用特征数的对数。整数或浮点数：指定特征数或比例。

8. random_state （默认 None）：控制随机性的参数。设置为固定整数可以让结果可复现。

9. max_leaf_nodes （默认 None）：限制树的叶子节点数。如果设置了，则会优先生成“最佳”的叶子，直到达到上限。

10. min_impurity_decrease （默认 0.0）：节点分裂所需的最小不纯度减少量。如果分裂不能带来至少该值的 impurity 减少，则不会继续分裂。

11. class_weight （默认 None）：给不同类别分配权重，用于处理类别不平衡问题。可选值："balanced"：自动根据样本数量调整权重。字典：手动指定权重，例如 {0:1, 1:5}。

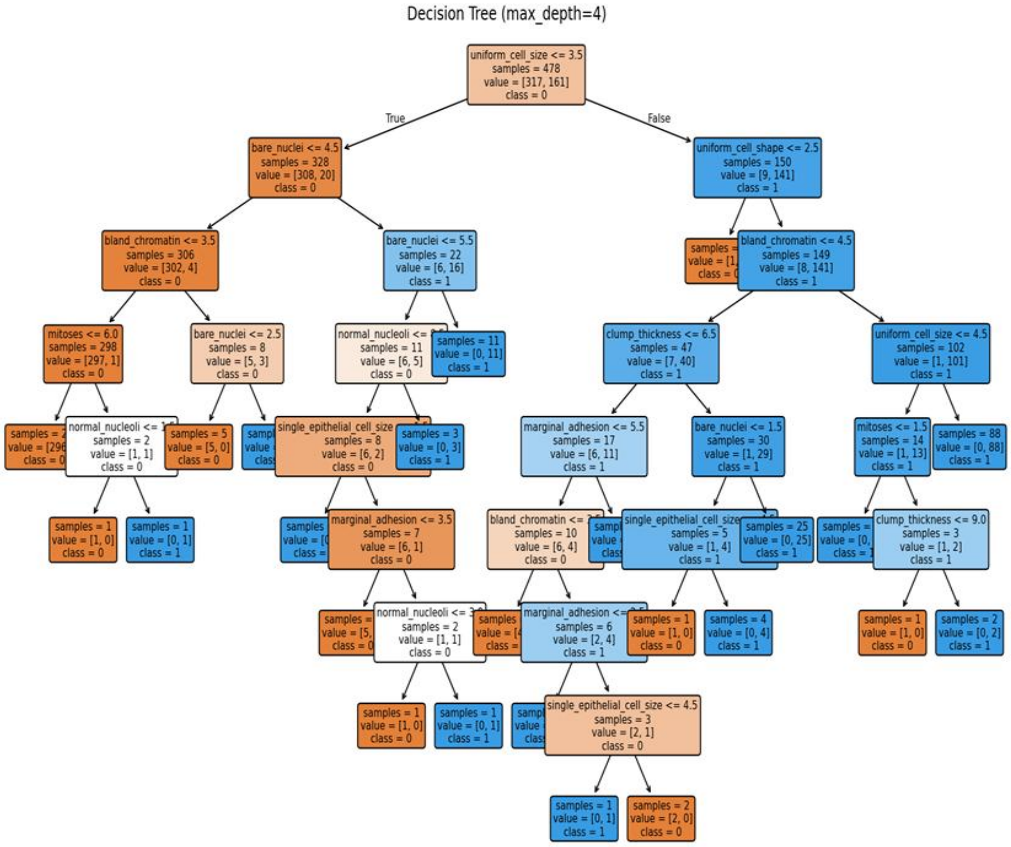
12. ccp_alpha （默认 0.0）：用于 最小代价复杂度剪枝 (Minimal Cost-Complexity Pruning)。

plot_tree()是 scikit-learn 提供的可视化函数，通过 from sklearn.tree import plot_tree 导入后可以直接使用，具体参数含义可以看下面的示例。下面是对肿瘤数据进行决策树建模，使用 70%的样本数据进行训练，30%的样本数据进行检验，然后给出模型效果评估的混淆矩阵。具体代码和运行结果如下：

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt
# 1. 读取数据 (请替换成你实际存放路径)
import readDataFromWugangTOP as wt
url = "https://wugang69.top/dataset/breast.csv"
breast = wt.read_csv_with_headers(url)
breast.replace('NA', np.nan, inplace=True)
# 删除缺失值
breast.dropna(inplace=True)
# 将 bare_nuclei 列转换为整数
breast['bare_nuclei'] = breast['bare_nuclei'].astype(int)
# 分离特征和标签
X = breast.drop(columns=["id","class"])
y = breast["class"].map({2:0, 4:1})   # 0=良性, 1=恶性
# 4. 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# 2. 训练决策树 (max_depth=4)
clf = DecisionTreeClassifier(criterion="gini", random_state=42)
clf.fit(X_train, y_train)
# 3. 预测并输出混淆矩阵
y_pred = clf.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print("混淆矩阵：")
print(cm)
print("\n分类结果：")
print(classification_report(y_test, y_pred, digits=4))
plt.figure(figsize=(16, 10))
plot_tree(
    clf,  # 决策树模型
    feature_names=X.columns, # 加上指标名称
    class_names=[str(c) for c in clf.classes_],
    filled=True,           # 保留色彩
    rounded=True,          # 圆角
    impurity=False,        # 去掉 gini
    proportion=False,      # 不显示比例
    precision=2,
)
plt.title("Decision Tree (max_depth=4)")
plt.show()
```

运行结果如下所示。可以观察到分类精度大约为 95%，有 4 个良性肿瘤误分类成恶性，有 7 个恶性肿瘤误分类为良性。

```
混淆矩阵：
                          预测良性        预测恶性
            真实良性          [[123             4]
            真实恶性           [  7            71]]
分类结果：
                precision    recall  f1-score   support

             0     0.9462    0.9685    0.9572       127
             1     0.9467    0.9103    0.9281        78

    accuracy                         0.9463       205
   macro avg     0.9464    0.9394    0.9427       205
weighted avg     0.9463    0.9463    0.9461       205
```



如果完全根据决策树算法对数据分裂，生成的决策树就是很大的，而且会造成过拟合，因此需要对决策树进行剪枝，Python 语言对决策数据剪枝常常使用函数 cost_complexity_pruning_path()获得一系列可选择的α值，即 ccp_alpha，这个值全称是 Complexity Parameter α。获取最佳复杂度参数的通常做法是

(1)先训练一棵大树；(2)调用 cost_complexity_pruning_path() 得到候选的 ccp_alpha；(3)用交叉验证挑选最优的 ccp_alpha；(4)重新训练最终模型。具体实现语句如下：

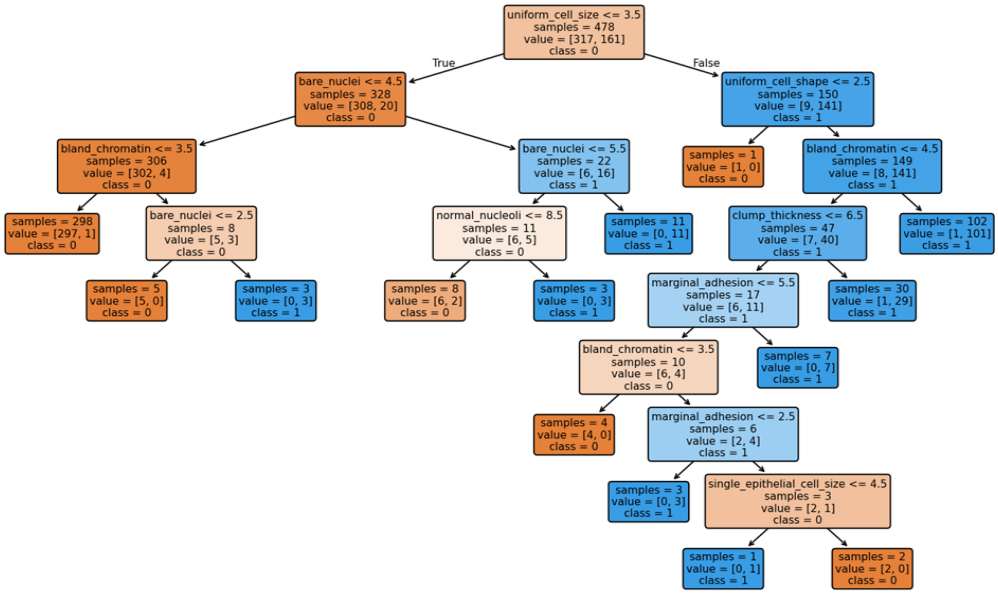
```
from sklearn.model_selection import train_test_split, cross_val_score
path = clf.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas = path.ccp_alphas

# 5. 用交叉验证选择最佳 alpha
cv_scores = []
for alpha in ccp_alphas:
    clf_temp = DecisionTreeClassifier(random_state=42, ccp_alpha=alpha)
    scores = cross_val_score(clf_temp, X_train, y_train, cv=5)
    cv_scores.append(np.mean(scores))

# 找到分数最高的 alpha
best_alpha = ccp_alphas[np.argmax(cv_scores)]
print("最佳 ccp_alpha:", best_alpha)

# 6. 使用最佳 alpha 重新训练
pruned_clf = DecisionTreeClassifier(random_state=42, ccp_alpha=best_alpha)
pruned_clf.fit(X_train, y_train)

# 7. 预测并输出混淆矩阵
y_pred = pruned_clf.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print("剪枝后的混淆矩阵：\n", cm)

# 8. 绘制简化决策树（只保留特征名和类别名）
plt.figure(figsize=(14, 8))
plot_tree(
    pruned_clf,
    feature_names=X.columns,
```

```
        class_names=[str(c) for c in clf.classes_],
        filled=True,
        rounded=True,
        fontsize=8,
        impurity=False,
        proportion=False
    )
    plt.show()
```

运行的结果如下：

```
最佳 ccp_alpha: 0.0068
剪枝后的混淆矩阵：
 [[124   3]
 [  8  70]]
```

使用上面的方法，获得的最佳 ccp_alpha 为 0.0068。对于每个非叶子节点，计算其 α 值，如果该节点的 α 值 ≤ best_alpha，就剪掉该节点的子树，这个过程从底向上递归进行，最后得到剪枝后的决策树，下图就是具体剪枝后的决策树



另外该决策树模型的预测准确率不如上一节的神经网络预测准确率高，这就说明不同的算法对不同的样本，有效性是不一样的，这正是研究的价值，也是提出很多各种分类算法的原因。

第三节 综合实验十二

一、实验目的

1. 掌握神经网络模型实现给定样本的预测
2. 掌握基于决策树模型的数据分类和预测

二、实验平台和软件

1. 互联网平台
2. 计算机系统
3. Spyde 环境和 Python 语言编译系统

三、实验内容和步骤

1. 神经网络建模

(1) 根据实验指导，读取乳腺癌数据集

(2) 生成乳腺癌 dataframe 数据结构

(3) 取生成的乳腺癌数据集中 70%样本为训练集，30%样本为验证集合

(4) 使用神经网络 Kreas 中的函数进行模型训练

(5) 使用构建的神经网络模型对验证样本集进行预测

(6) 计算该模型在样本验证集上的分类准确率

2. 决策树建模

(1) 使用上面的乳腺癌数据进行决策树建模

(2) 先进行最大分裂为 4 层的决策树训练，检验模型准确率

(3) 实现对生成的决策树剪枝

(4) 绘制剪枝后的决策树图形

(5) 使用优化的决策树模型对验证变量进行预测计算

(6) 计算该决策树模型在验证集上的分类准确率

四、实验报告（总分 8 分）

1.使用 creditData 数据集，抽取 70%和 30%分别作为训练集和验证集，数据集中第 15 列为类别变量，使用其它变量进行神经网络训练，预测验证集，给出验证结果和精确度估计。说明：这是一个信用卡数据集，下载文档说明见下面的链接，数据下载：https://wugang69.top/dataset/creditData.csv。(3 分)。

http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/australian/australian.doc

2.对于 iris 数据集，使用决策树建模，70%为训练集，30%为验证集。使用决策树建模，然后分析剪枝，生成最优决策树用来测试验证集，给出模型准确率。 (3 分)

3.记录实验过程，分析实验问题和解决办法。 (2 分)