# 《机器学习与深度学习》课程

# 实 验 报 告



**姓 名： 金家耀**

**专 业：**  人工智能

**学 号： 1193210320**

**江南大学人工智能与计算机学院**

# 集成学习

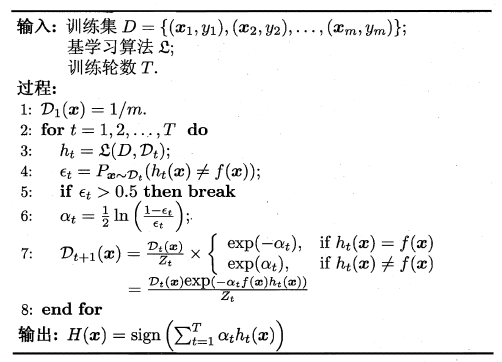
**1实验目的**

集成学习(ensemble learning) 通过构建并结合多个学习器来完成学习任务，常可获得比单一学习器显著优越的泛化性能。集成学习类型之一的Boosting是个体学习器间存在强依赖关系、串行生成的序列化方法。AdaBoost是Boosting 族算法最著名的代表。本实验的目的在于加深学生对AdaBoost 算法的理解，掌握算法的实现过程和使用方法。

**2实验原理**

AdaBoost的工作机制是:先从初始训练集训练出一个基学习器，再根据基学习器的表现对训练样本分布进行调整，使得先前基学习器做错的训练样本在后续受到更多关注，然后基于调整后的样本分布来训练下一个基学习器;如此重复进行，直至基学习器数目达到事先指定的值T， 最终将这T个基学习器进行加权结合。

AdaBoost算法的代码如下：



**3实验内容和要求**

疝病是描述马胃肠痛的术语。然而，这种病不一定源自马的胃肠问题，其他问题也可能引发马疝病。我们提供的马氙气数据集中包含了医院检测马疝病的一些指标和马的状态标签，我们要做的就是通过AdaBoost算法训练299个样本，形成一个以决策树为基学习器的集成学习器，并对67个样本进行测试。

horseColicTraining.txt为训练数据集，共299条训练数据，每条数据前21个为特征，最后一个为马的状况标签：1—仍存活，0—未能存活。

horseColicTest.txt为测试数据集，共67条测试数据，每条数据前21个为特征，最后一个为马的状况标签：1—仍存活，0—未能存活。

实验具体要求：

1）使用 Python 语言实现 AdaBoost 算法，在马氙气数据集（horseColicTest.txt，horseColicTraining.txt）上训练一个集成分类器，估计马疝气的死亡率。

2）画图输出混淆矩阵

3）计算并给出查准率、查全率和 F1 度量

4）绘制 P-R 曲线。

**4实验代码和结果**

代码已上传至<https://github.com/shinejjy/MachineLearningAndDeepLearning/sy4>

1. 读取训练集和测试集

import pandas as pd  
import numpy as np  
from sklearn.tree import DecisionTreeClassifier  
import copy  
  
# 读入训练数据集和测试数据集  
train = pd.read\_csv('horseColicTraining.txt', delimiter='\t', header=None)  
test = pd.read\_csv('horseColicTest.txt', delimiter='\t', header=None)  
# 划分特征和标签  
X\_train, y\_train = train.iloc[:, :21].to\_numpy(), train.iloc[:, 21].to\_numpy()  
X\_test, y\_test = test.iloc[:, :21].to\_numpy(), test.iloc[:, 21].to\_numpy()  
# 将标签为0的样本改为-1  
y\_train[y\_train == 0] = -1  
y\_test[y\_test == 0] = -1  
  
print(X\_train.shape)  
print(y\_train.shape)  
print(X\_test.shape)  
print(y\_test.shape)

首先，通过pd.read\_csv函数读取了两个文本文件（'horseColicTraining.txt'和'horseColicTest.txt'），这两个文件包含了关于马的生物医学特征的数据。在数据加载后，通过iloc方法将特征和标签分别划分为训练集（X\_train和y\_train）和测试集（X\_test和y\_test）。

接下来，标签数据中的0值被转换为-1，这是因为在adaboost问题中调节样本比重时，通常将类别标签设定为1和-1。

1. 构建基分类器为底层决策树的AdaBoost分类器

class AdaBoostClassifier:  
 def \_\_init\_\_(self, n\_estimators=20, base\_clf=DecisionTreeClassifier(max\_depth=2)):  
 *"""* ***:param*** *n\_estimators: 基分类器个数* ***:param*** *base\_clf: 基分类器种类  
 """* self.n\_estimators = n\_estimators  
 self.alphas = []  
 self.classifiers = []  
 self.base\_clf = base\_clf  
  
 def fit(self, X\_train, y\_train):  
 m, n = X\_train.shape  
 w = np.ones(m) / m # 归一化  
  
 for i in range(self.n\_estimators):  
 base\_classifier = copy.deepcopy(self.base\_clf)  
 base\_classifier.fit(X\_train, y\_train, sample\_weight=w) # 用特定的权重进行训练  
 y\_pred = base\_classifier.predict(X\_train) # 预测训练样本  
  
 error\_o = y\_pred != y\_train # 计算错误率  
 error = np.sum(w \* error\_o) # 计算加权后的错误率  
 print('base\_clf\_%02d train acc: %.4f'%(i + 1, 1 - np.sum(error\_o) / len(X\_train)))  
 alpha = 0.5 \* np.log((1 - error) / error) # 设置基学习器权重  
 w = w \* np.exp(-alpha \* y\_train \* y\_pred) # 更新样本比重  
 w = w / np.sum(w) # 归一化样本  
  
 self.alphas.append(alpha)  
 self.classifiers.append(base\_classifier)  
  
 def predict(self, X\_test):  
 predictions = np.zeros(X\_test.shape[0])  
  
 for alpha, classifier in zip(self.alphas, self.classifiers):  
 predictions += alpha \* classifier.predict(X\_test) # 求和各学习器加权预测值  
  
 return np.sign(predictions)

ad\_boost = AdaBoostClassifier(n\_estimators=20, base\_clf=DecisionTreeClassifier(max\_depth=2, random\_state=42))  
ad\_boost.fit(X\_train, y\_train)

1. 死亡率估计

通过AdaBoost算法在测试集上的预测，我们估计马疝气的死亡率为：

## 1) 计算死亡率  
y\_pred = ad\_boost.predict(X\_test)  
print('Rate of death is: ', np.sum(y\_pred == -1) / len(y\_pred))

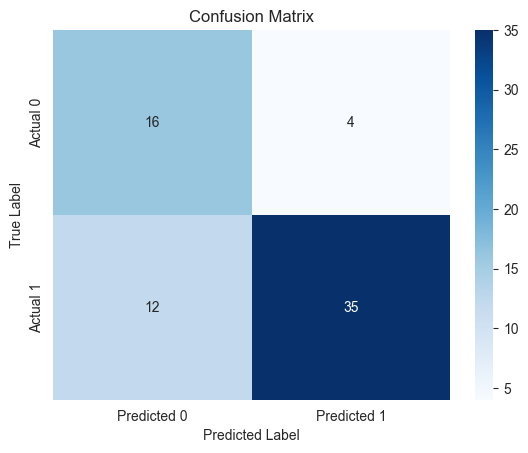
这一指标反映了模型对于测试集中马疝气致命状况的预测性能。在实验中，我们观察到模型的死亡率估计是关键的评估指标之一，尤其是对于这类医学问题。

1. 混淆矩阵可视化

混淆矩阵是一种直观的方式，用于查看模型在不同类别上的分类性能。我们使用seaborn库绘制混淆矩阵的热图：

from matplotlib import pyplot as plt  
from sklearn.metrics import confusion\_matrix, precision\_score, recall\_score, f1\_score, precision\_recall\_curve, auc  
import seaborn as sns  
  
## 2) 画图输出混淆矩阵  
conf\_matrix = confusion\_matrix(y\_test, y\_pred)  
sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Predicted 0', 'Predicted 1'], yticklabels=['Actual 0', 'Actual 1'])  
plt.xlabel('Predicted Label')  
plt.ylabel('True Label')  
plt.title('Confusion Matrix')  
plt.show()

混淆矩阵的四个象限分别对应模型的真正例、真负例、假正例和假负例。通过该可视化，我们能够直观地了解模型在不同类别上的表现。



1. 性能指标计算

我们计算了查准率、查全率和F1度量等性能指标，提供了对模型不同方面性能的更详细了解：

# 3) 计算查准率、查全率和 F1 度量  
precision = precision\_score(y\_test, y\_pred)  
recall = recall\_score(y\_test, y\_pred)  
f1 = f1\_score(y\_test, y\_pred)  
  
print(f'Precision: {precision:.4f}')  
print(f'Recall: {recall:.4f}')  
print(f'F1 Score: {f1:.4f}')

·查准率（Precision）： 模型在预测为正类别的样本中，有多少是真正的正类别。

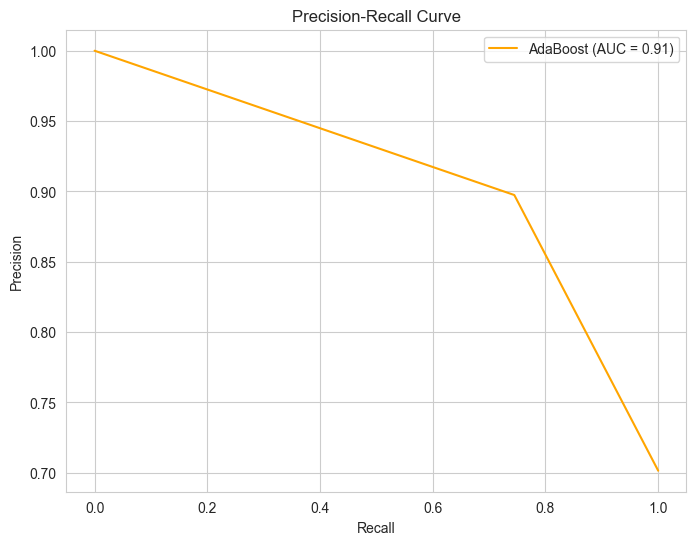
·查全率（Recall）： 模型在所有真正的正类别样本中，有多少被成功预测为正类别。

·F1 Score： 综合了查准率和查全率，是一个平衡指标，适用于不同类别分布的情况。

1. P-R 曲线

P-R 曲线提供了在不同阈值下模型性能的详细信息。我们计算了P-R曲线并计算了曲线下的面积（AUC）：

# 4) 绘制 P-R 曲线  
precision\_curve, recall\_curve, \_ = precision\_recall\_curve(y\_test, ad\_boost.predict(X\_test))  
area\_under\_curve = auc(recall\_curve, precision\_curve)  
  
plt.figure(figsize=(8, 6))  
plt.plot(recall\_curve, precision\_curve, label=f'AdaBoost (AUC = {area\_under\_curve:.2f})', color='orange')  
plt.xlabel('Recall')  
plt.ylabel('Precision')  
plt.title('Precision-Recall Curve')  
plt.legend()  
plt.show()



P-R曲线下的面积提供了一个综合性的性能度量，能够更全面地评估模型在查准率和查全率之间的平衡。