

**2024年《机器学习》**

**工程报告**

**（完整版）**



**课 程：** 机器学习

**姓名+学号：** 孙赫阳2022217450

**姓名+学号：** 郑豪2022217456 、杜金浩2022217447

黄鑫2022217483 、宁子扬2022217460

**完成时间：** 5月19日

一． 工程摘要与每人贡献

摘要：总体介绍本工程的工作。  
数据集链接：[League of Legends Turkiye SoloQ Matches: M-GM-CHL (kaggle.com)](https://www.kaggle.com/datasets/huseyinbattal/league-of-legends-turkiye-soloq-matches-m-gm-chl)

GitHub链接：[GitHub - river-and-wood/machineLearning2024: HFUT machine learning 2024 report](https://github.com/river-and-wood/machineLearning2024)

每人分工：

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 角色  （组长，组员，独自完成） | 工作量  比例 | 负责内容 |
| 孙赫阳 | 组长 | 20% | 完成报告中第三部分、模型方法的撰写 |
| 宁子扬 | 组员 | 20% | 实验结果分析，对比和讨论 |
| 郑豪 | 组员 | 30% | 数据集收集，代码撰写，报告汇总 |
| 杜金浩 | 组员 | 15% | 大作业报告中的第四部分系统设计 |
| 黄鑫 | 组员 | 15% | 研究背景与意义，对本科的感想，建议 |

二、研究背景与意义

在当今社会，电子竞技已经成为一个备受瞩目的产业。作为全球最受欢迎的电子竞技游戏之一，英雄联盟（League of Legends，简称LOL）在全球范围内拥有庞大的玩家基础和观众群体。随着电子竞技行业的蓬勃发展，数据分析和机器学习技术在电竞产业中的应用也越来越广泛。

本研究的背景与意义在于，通过对英雄联盟电竞比赛数据进行分析和挖掘，利用Scikit-learn、TensorFlow或PyTorch等机器学习软件包，探索不同的机器学习的模型与方法在电竞比赛数据分类中的应用效果。这将为电竞战队、选手和教练提供有价值的比赛策略和训练指导，从而提高比赛成绩和竞技水平。

本研究将使用的数据集为上传的CSV文件，其中包含了英雄联盟电竞比赛的相关数据。通过分析这些数据，我们可以了解到不同战队、选手和英雄的竞技表现，进而为机器学习模型的训练和评估提供基础。

在机器学习模型的选择上，我们组将尝试使用决策树、逻辑回归、支持向量机、神经网络、随机森林和部分依赖图等多种方法，并将在下个部分详细描述与分类。这些模型在电竞比赛数据分类中具有不同的特点和应用场景，通过对它们的比较和评估，我们可以找到最适合英雄联盟电竞比赛数据分类的模型。

此外，本研究还将对机器学习模型进行可视化展示，以便更直观地了解不同模型在分类任务中的表现。通过可视化，我们可以观察到模型在不同特征上的权重和重要性，从而为后续的模型优化和特征选择提供依据。

本研究旨在通过对英雄联盟电竞比赛数据的挖掘和分析，探索机器学习技术在电竞比赛分类中的应用价值。这将为电竞行业的发展提供有力支持，同时也为机器学习技术在电竞领域的应用提供新的思路和方法，而对于我们在机器学习的探究道路上也具有一定的指导意义与影响。

1. 模型方法

首先我们代码使用了pandas库来处理数据：

* 使用了sklearn库中的train\_test\_split进行数据集的划分。
* 使用了GradientBoostingClassifier作为分类模型。
* 评估模型的性能时使用了accuracy\_score, classification\_report, 和 confusion\_matrix。
* 使用了PartialDependenceDisplay来展示特征对模型预测的影响。
* 使用了matplotlib和seaborn库进行数据可视化。

所使用的模型：

1. **梯度提升分类器（Gradient Boosting Classifier）**：
   1. 工程使用了梯度提升分类器作为主要的机器学习模型。
   2. 梯度提升分类器是一种基于决策树的集成学习方法，通过逐步构建模型来最小化损失函数，从而达到更好的性能。
   3. 在这个模型中，使用了100棵树，学习率为0.1，并且每棵树的最大深度为3。
2. **逻辑回归模型（Logistic Regression）**：
   1. 除了梯度提升分类器，工程还训练了一个逻辑回归模型。
   2. 逻辑回归是一种线性分类方法，通过最大似然估计来估计模型的参数。
   3. 在这个模型中，使用了标准化后的数据进行训练。
3. **决策树（Decision Tree）**：
   1. 文本中提到了一个限制性决策树的可视化。
   2. 决策树是一种简单的非参数模型，用于分类和回归。
4. **支持向量机（Support Vector Machine, SVM）**：
   1. 工程中还使用了支持向量机作为分类器。
   2. SVM是一种强大的机器学习模型，用于分类和回归。
   3. 在这个模型中，使用了线性核，并且C值为0.000003。
5. **随机森林（Random forest）**：

a. n\_estimators=100：表示随机森林中包含100棵决策树。

b. random\_state=42：表示随机森林的随机种子设置为42，用于保持实验的可重复性。

1. **逻辑回归（Logistic Regression）**：

a. 代码中使用了逻辑回归模型作为分类器。

b.代码中提到了特征选择、目标变量转换、模型训练、预测、准确率计算、混淆矩阵生成和分类报告生成等步骤。

1. **部分依赖图（Partial Dependence Plot, PDP）**：代码中使用了PartialDependenceDisplay类来生成部分依赖图。
2. **神经网络（Neural Network）**：

。在MLP模型中，我们使用了ReLU激活函数，这是一种常用的激活函数，它将输入数据的负值设置为0，从而增加了模型的非线性表达能力。

同时，我们设置了学习率为0.001，这是一个较小的值，意味着模型对参数的调整较小，可能会需要更长的训练时间来达到收敛。

这些模型的优缺点：

1. **梯度提升分类器**：**优点**：能够处理高维数据，具有较好的泛化能力。
   1. **缺点**：可能需要较长的训练时间，对特征选择和超参数调整敏感。
2. **逻辑回归模型**：**优点**：简单、易于理解和实现，对线性关系拟合效果好。
   1. **缺点**：假设数据是线性可分的，对非线性关系拟合效果不佳。
3. **决策树**：**优点**：易于理解和解释，计算效率高。
   1. **缺点**：容易过拟合，对噪声数据敏感。
4. **支持向量机**：**优点**：对高维数据有较好的性能，可以处理非线性问题。
   1. **缺点**：对参数C的选择敏感，可能需要较长的训练时间。
5. **随机森林**：**优点**：具有较高的准确率，可以处理高维数据，不易过拟合。
   1. **缺点**：对特征选择和超参数调整敏感。
6. **神经网络**：**优点**：强大的表示能力和自适应特性，适用于复杂的非线性问题。
   1. **缺点**：需要大量的数据和计算资源，训练时间长，解释性较差。
7. **部分依赖图**：**优点**：可视化模型对特征的依赖关系，帮助理解模型。
   1. **缺点**：无法显示整个数据集的依赖关系

各个模型的理论基础：

**1.梯度提升分类器：**

* **理论基础**：基于前向分步算法（如CART算法），通过迭代地构建一系列的决策树来最小化损失函数。每个决策树都试图修正前一个树的错误，从而提高整体模型的性能。

1. **逻辑回归模型：**

* **理论基础**：基于最大似然估计（Maximum Likelihood Estimation, MLE），通过最小化似然函数来估计模型的参数。似然函数是模型输出与真实值之间的概率。

1. **决策树：**

* **理论基础**：基于贪心算法，通过递归地将数据集分割成子集，直到满足停止条件（如最大深度限制或信息增益小于阈值）。

1. **支持向量机：**

* **理论基础**：基于结构风险最小化原则，通过最大化决策边界（分类问题）或最小化回归平方误差（回归问题）来找到最优的模型。

1. **随机森林：**

* **理论基础**：基于集成学习方法，通过构建多个决策树，并将它们的预测结果取平均值或投票来提高模型的性能。

1. **神经网络：**

* **理论基础**：基于多层感知机（Multilayer Perceptron, MLP）模型，通过模拟人脑神经元之间的连接方式，构建非线性的复杂模型。

1. **部份依赖图：**

* **理论基础**：基于模型在特征上的局部依赖性，通过计算模型对每个特征的局部依赖性来可视化特征对预测结果的影响。

四、系统设计

2022217456 郑豪

1. 系统流程概述

数据读取与预处理：从CSV文件中读取数据，并进行初步的数据检查和预处理。

特征选择：选择与目标变量（‘won’）相关的特征。

数据清洗：处理缺失值，确保所有特征都是数值型。

数据集划分：将数据集划分为训练集和测试集。

模型训练：使用梯度提升分类器（Gradient Boosting Classifier）进行模型训练。

模型评估：在测试集上评估模型的性能，并生成各种评估报告。

可视化：绘制特征重要性、混淆矩阵、箱线图、直方图、相关性矩阵和部分依赖图。

2. 详细流程

2.1 数据读取与预处理

读取CSV文件。

打印列名，检查数据的基本结构。

2.2 特征选择

选择与目标变量（‘won’）相关的特征。

2.3 数据清洗

处理缺失值：删除包含缺失值的行。

确保所有特征都是数值型：将特征转换为数值型，并再次处理缺失值。

将目标列转换为数值型。

2.4 数据集划分

使用train\_test\_split函数将数据集划分为训练集和测试集。

2.5 模型训练

使用梯度提升分类器（Gradient Boosting Classifier）进行模型训练。

2.6 模型评估

在测试集上评估模型的性能。

打印准确率、分类报告和混淆矩阵。

2.7 可视化

绘制特征重要性、混淆矩阵、箱线图、直方图、相关性矩阵和部分依赖图。

2022217450 孙赫阳

1. 系统流程概述

数据读取与预处理：从CSV文件中读取数据，并进行初步的数据检查和预处理。

特征转换：将非数值型特征转换为数值型特征。

数据集划分：将数据集划分为训练集和测试集。

模型训练：使用决策树分类器进行模型训练。

模型评估：在测试集上评估模型的性能，并生成评估报告。

可视化：绘制决策树的可视化图形。

2. 详细流程

2.1 数据读取与预处理

读取CSV文件。

打印数据的前几行，以检查数据的基本结构。

检查数据中的缺失值，并打印包含缺失值的列的数量。

2.2 特征转换

将目标变量（‘won’）转换为数值型特征。

识别数据集中的非数值型列。

使用LabelEncoder将非数值型列转换为数值型特征。

2.3 数据集划分

使用train\_test\_split函数将数据集划分为训练集和测试集。

2.4 模型训练

使用决策树分类器进行模型训练。

2.5 模型评估

在测试集上评估模型的性能。

打印准确率和分类报告。

2.6 可视化

绘制决策树的可视化图形。

2022217447杜金浩

1. 系统流程概述

数据读取与预处理：从CSV文件中读取数据，并进行初步的数据检查和预处理。

特征转换与标准化：将非数值型特征转换为数值型特征，并对特征进行标准化处理。

数据集划分：将数据集划分为训练集和测试集。

模型训练：使用支持向量机（SVM）分类器进行模型训练。

模型评估：在测试集上评估模型的性能，并生成评估报告。

可视化：绘制混淆矩阵的热力图。

2. 详细流程

2.1 数据读取与预处理

读取CSV文件。

打印数据的前几行，以检查数据的基本结构。

检查数据中的缺失值，并打印包含缺失值的列的数量。

2.2 特征转换与标准化

将目标变量（‘won’）与特征（X）分离。

使用LabelEncoder将非数值型特征转换为数值型特征。

使用StandardScaler对特征进行标准化处理。

2.3 数据集划分

使用train\_test\_split函数将数据集划分为训练集和测试集。

2.4 模型训练

使用支持向量机（SVM）分类器进行模型训练，其中C值设置为非常小的数（0.000003）。

2.5 模型评估

在测试集上评估模型的性能。

打印准确率和分类报告。

2.6 可视化

计算混淆矩阵。

绘制混淆矩阵的热力图，并设置中文字体。

2022217483黄鑫

1. 系统流程概述

数据读取与预处理：从CSV文件中读取数据，并进行初步的数据检查和预处理。

特征选择与编码：选择特征，并将非数值型特征转换为数值型特征。

数据集划分：将数据集划分为训练集和测试集。

模型训练：使用随机森林分类器进行模型训练。

模型评估：在测试集上评估模型的性能，并生成评估报告。

可视化：绘制特征重要性的条形图。

2. 详细流程

2.1 数据读取与预处理

读取CSV文件。

打印数据的前几行，以检查数据的基本结构。

检查数据中的缺失值，并打印包含缺失值的列的数量。

2.2 特征选择与编码

将目标变量（‘won’）与特征（X）分离。

使用LabelEncoder将非数值型特征转换为数值型特征。

2.3 数据集划分

使用train\_test\_split函数将数据集划分为训练集和测试集。

2.4 模型训练

使用随机森林分类器进行模型训练。

2.5 模型评估

在测试集上评估模型的性能。

打印准确率和分类报告。

2.6 可视化

获取特征重要性。

绘制特征重要性的条形图，并设置中文字体。

2022217460宁子杨

1. 系统流程概述

数据读取与预处理：从CSV文件中读取数据，并进行初步的数据检查和预处理。

特征选择：选择与目标变量（‘won’）相关的特征。

特征转换：将分类特征转换为数值型特征。

数据集划分：将数据集划分为训练集和测试集。

特征标准化：对特征进行标准化处理。

模型训练：使用逻辑回归模型进行模型训练。

模型评估：在测试集上评估模型的性能，并生成评估报告。

可视化：绘制混淆矩阵的热力图。

2. 详细流程

2.1 数据读取与预处理

读取CSV文件。

选择与目标变量相关的特征。

将分类特征转换为数值型特征。

2.2 数据集划分

使用train\_test\_split函数将数据集划分为训练集和测试集。

2.3 特征标准化

使用StandardScaler对特征进行标准化处理。

2.4 模型训练

使用逻辑回归模型进行模型训练。

2.5 模型评估

在测试集上评估模型的性能。

打印准确率、混淆矩阵和分类报告。

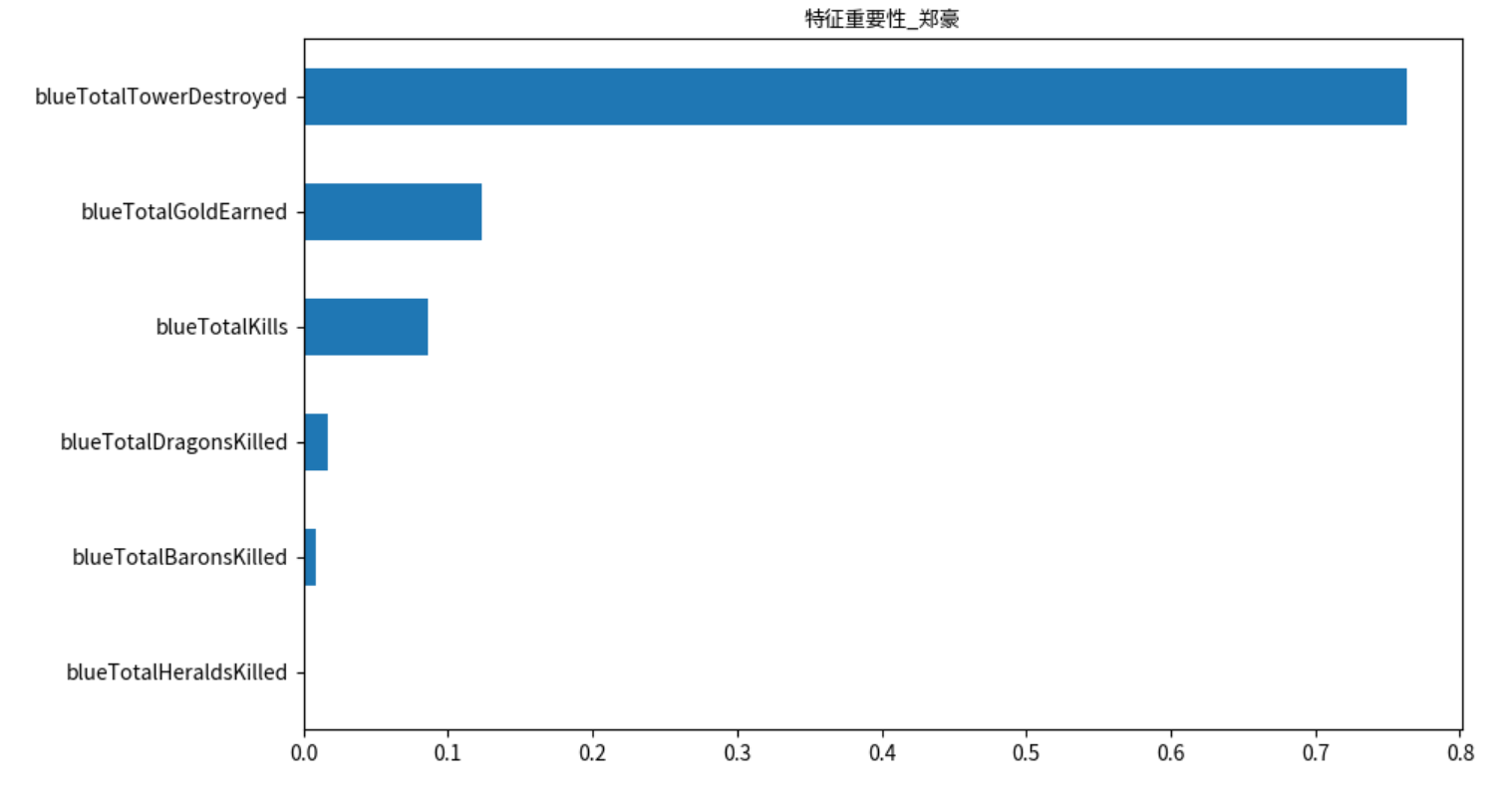
2.6 可视化

绘制混淆矩阵的热力图，并设置中文字体。

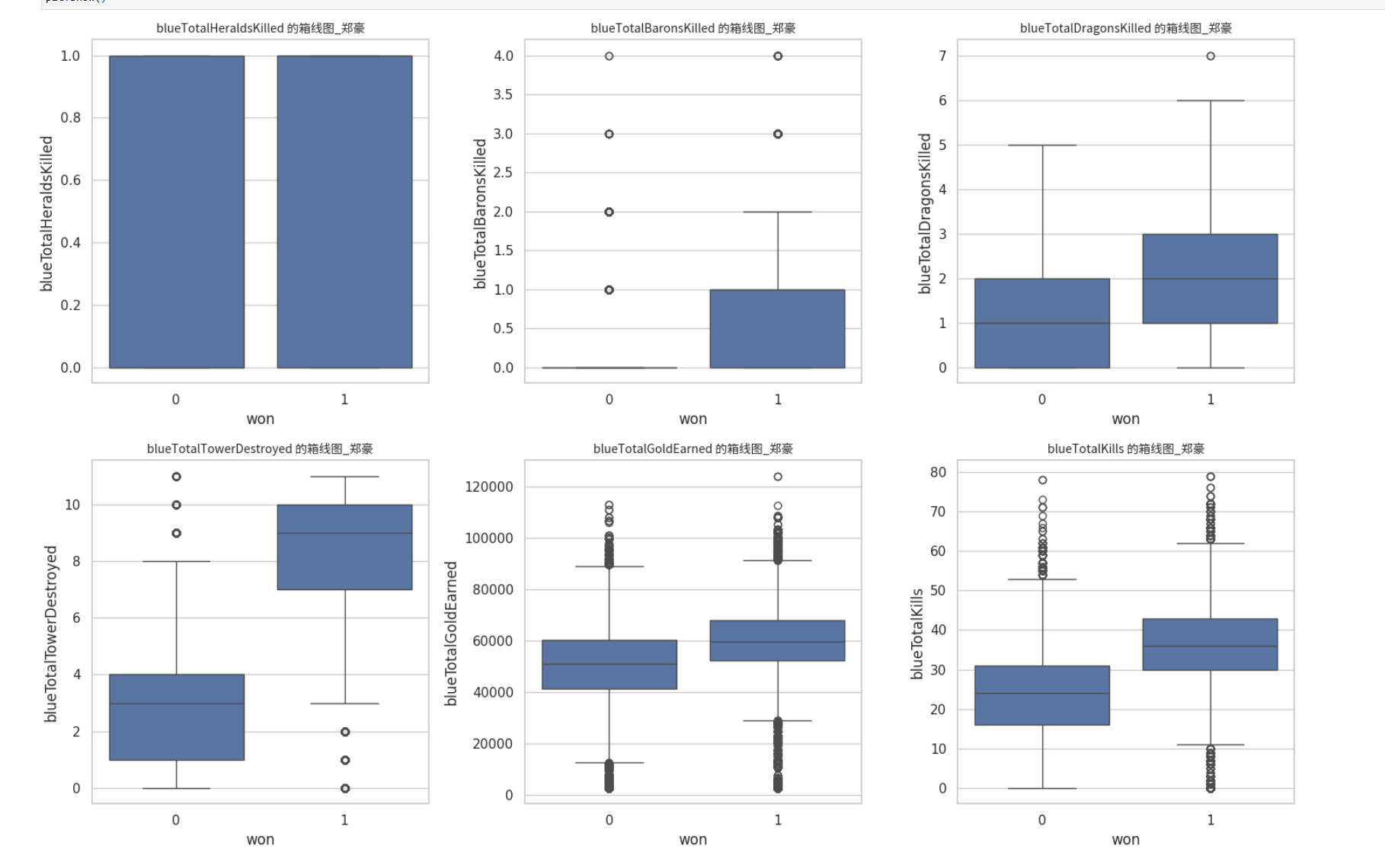
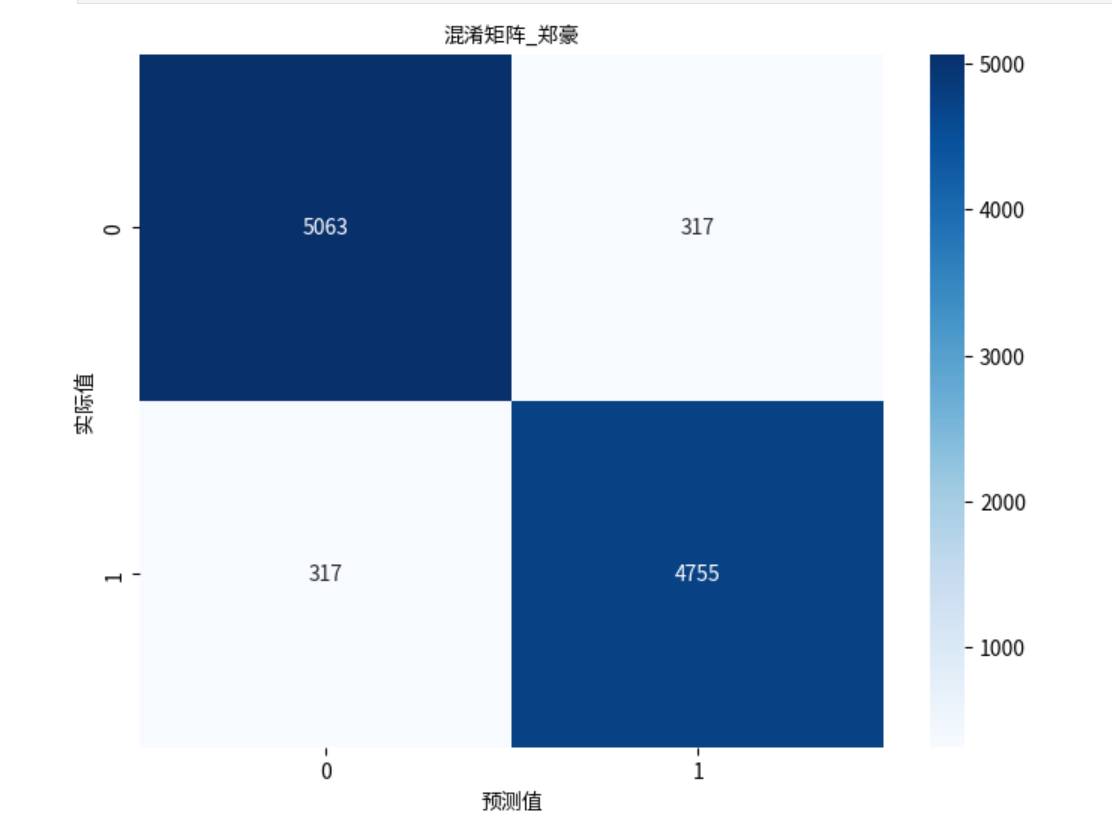
五．实验结果分析、对比和讨论

**郑豪**

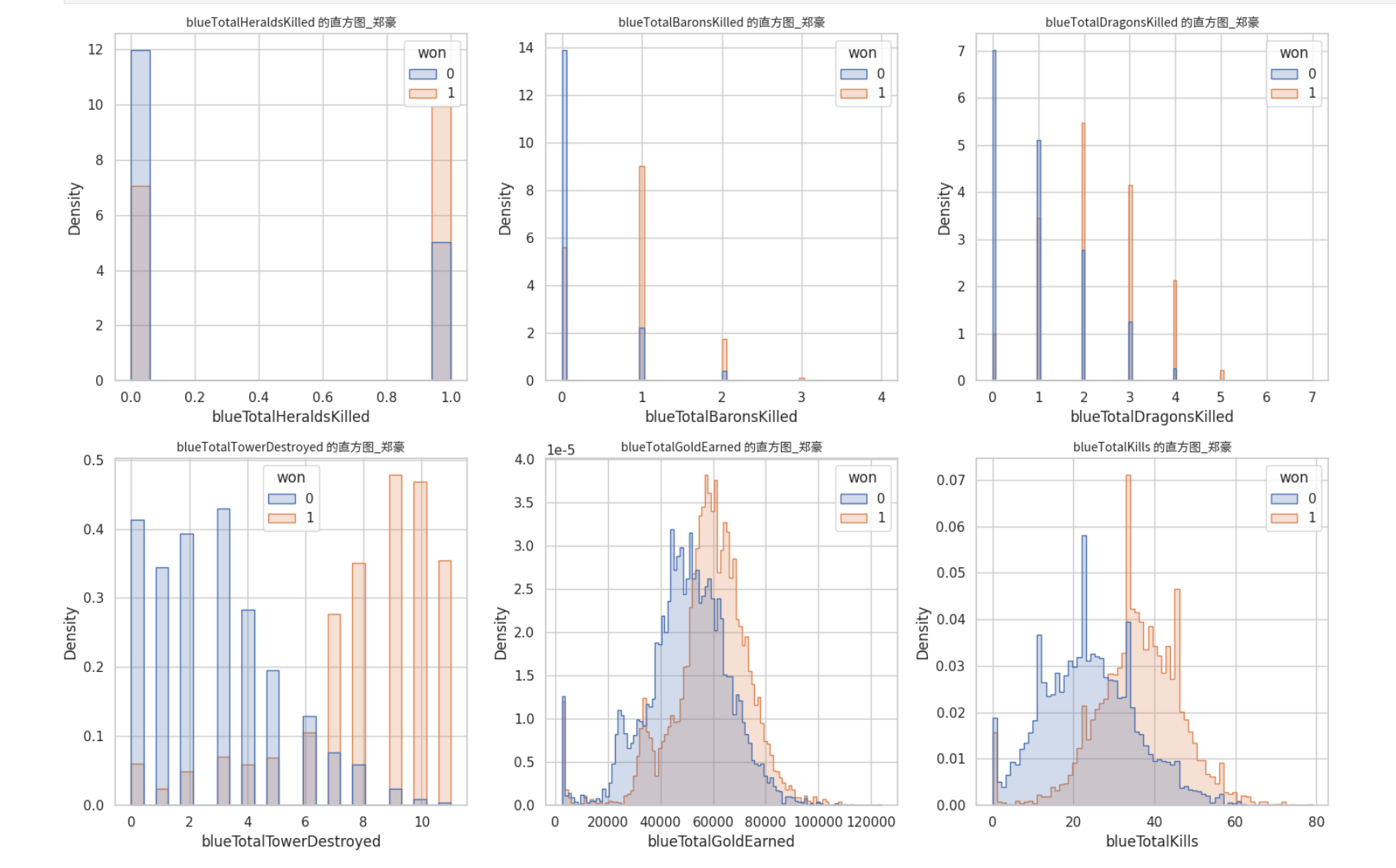
* 模型性能：准确率：梯度提升模型在测试集上的准确率为93.93%，这意味着模型能够正确预测比赛结果的概率非常高。
* 分类报告：精确度（precision）、召回率（recall）和F1分数都是评估分类模型性能的重要指标。从分类报告中可以看出，模型对于预测红队胜利（标签0）和蓝队胜利（标签1）的性能是均衡的，两者的精确度、召回率和F1分数都非常接近。
* 特征重要性：特征重要性图显示了各个特征对于模型预测结果的影响程度。从图中可以看出，某些特征（如蓝队总金币数）对模型预测的影响更大，而其他特征（如蓝队总杀死小龙数）的影响则较小。这可以帮助我们理解哪些游戏指标对于预测比赛结果最为关键。



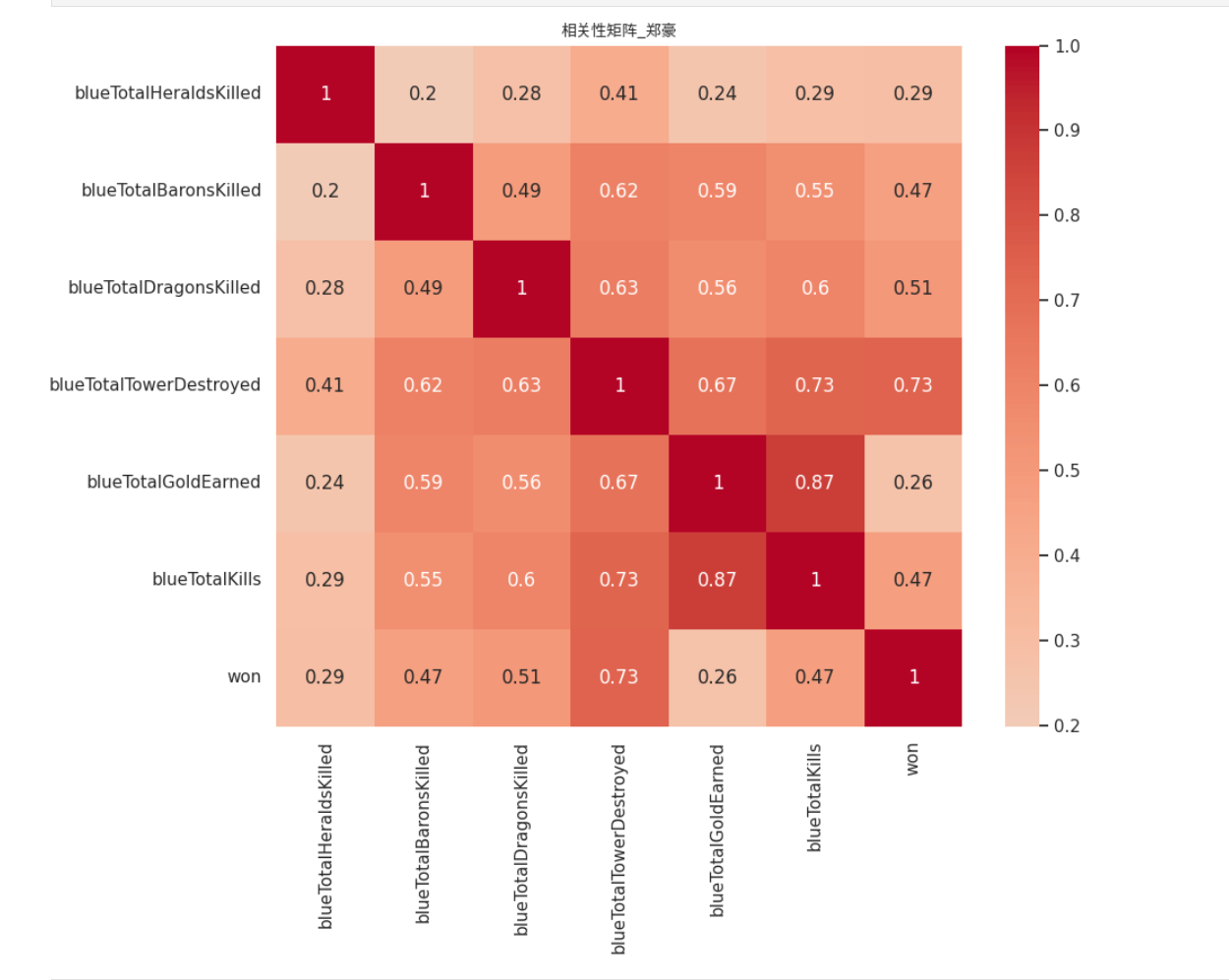
* 混淆矩阵：混淆矩阵提供了一个直观的方式来理解模型在各个类别上的预测性能。从混淆矩阵中可以看出，模型在预测红队胜利和蓝队胜利时的错误数量是相近的，这表明模型没有明显的偏向性。



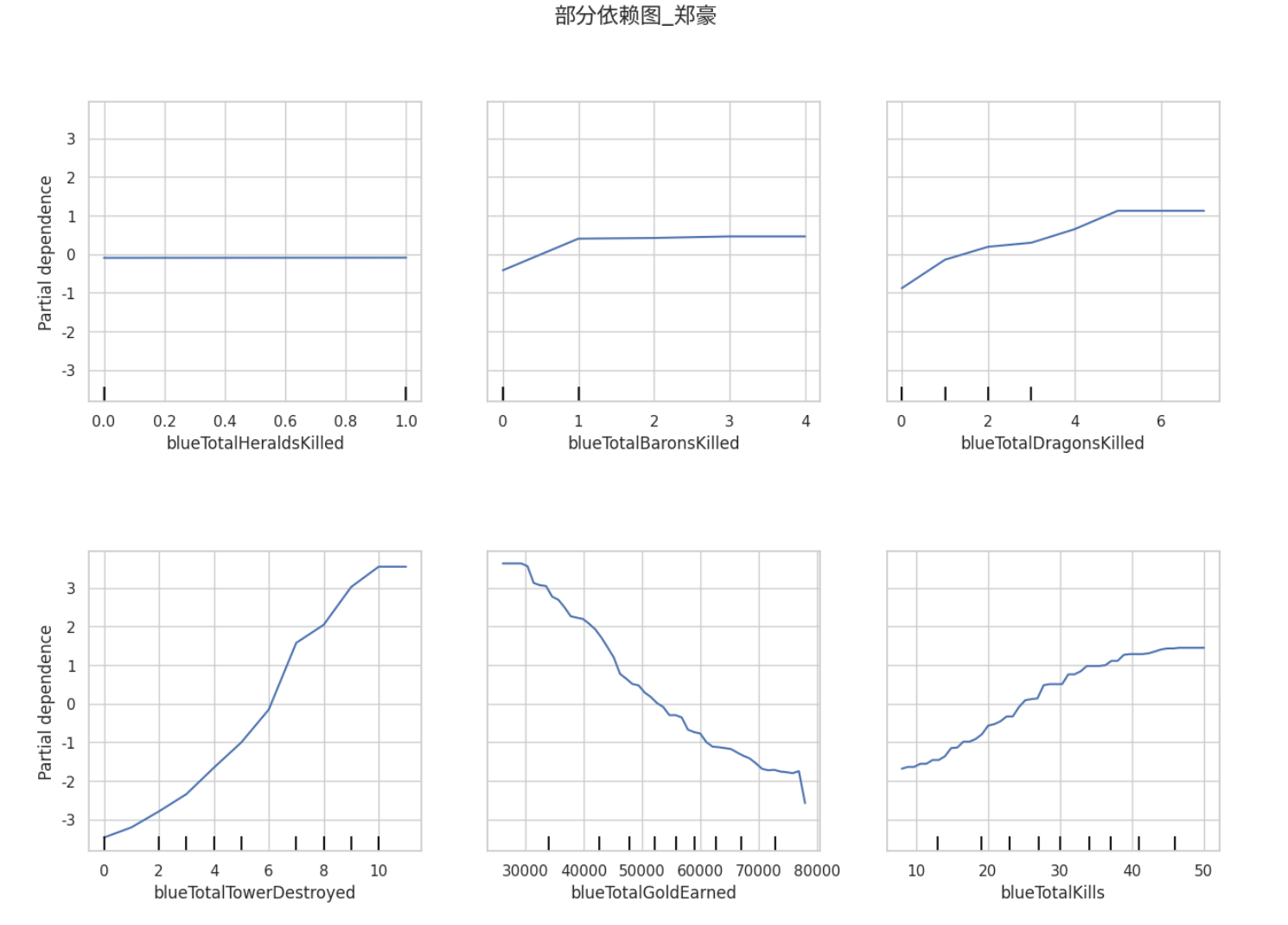
* 数据分布：箱线图和直方图展示了不同特征在不同比赛结果下的分布情况。这些图表可以帮助我们观察数据的分布特征，以及不同特征值与比赛结果之间的关系。



* 特征相关性：相关性矩阵和热力图显示了特征之间的相关性。高相关性可能表明某些特征之间存在冗余，这可能会影响模型的稳定性和预测能力。



* 部分依赖图（PDP）：PDP图显示了目标变量与特征之间的关系。通过PDP图，我们可以观察到随着某个特征值的增加，目标变量的平均值是如何变化的，这有助于我们理解模型是如何根据特定特征来预测结果的。



**孙赫阳**

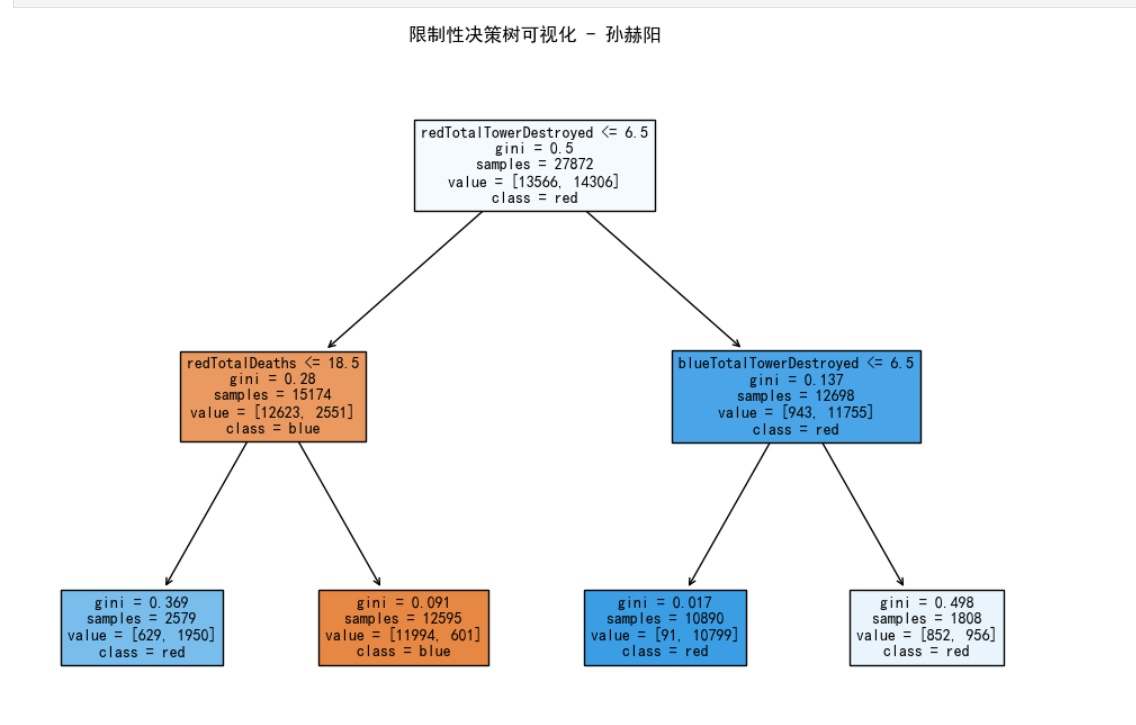
1. 数据集加载与预处理：数据集包含74列，其中包括游戏的各种统计数据，如比赛等级（matchElo）、游戏持续时间、游戏版本、各种游戏事件（如第一滴血、第一条龙等）以及两支队伍的各个英雄的统计数据。
   1. 数据集没有缺失值，这是通过isnull().sum()方法检查得出的。
   2. 目标变量'won'被编码为数值类型，其中1代表蓝队胜利，0代表红队胜利。
   3. 所有非数值列都被转换为数值类型，这可能是为了满足机器学习模型对输入数据的要求。
2. 数据集划分：数据集被划分为特征（X）和目标变量（y），然后进一步划分为训练集和测试集，其中测试集大小为数据集的20%。
3. 决策树可视化：

决策树的根节点为“redTotalTowerDestroyed <= 6.5”，其样本量为27872，基尼系数为0.5，价值为[13566, 14306]，类别为红色。该节点分为两个子节点：

* “redTotalDeaths <= 18.5”：样本量为15174，基尼系数为0.28，价值为[12623, 2551]，类别为蓝色。
* “blueTotalTowerDestroyed <= 6.5”：样本量为12698，基尼系数为0.137，价值为[943, 11755]，类别为红色。

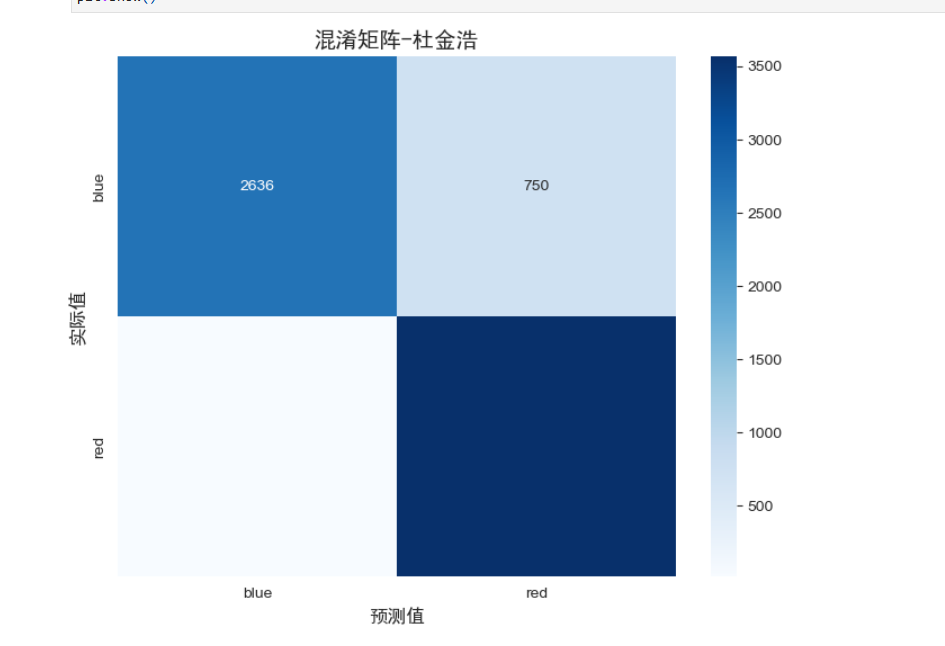
这两个子节点各自又分别分为两个更细小的节点：

* 对于“redTotalDeaths <= 18.5”节点：一个子节点的样本量为2579，基尼系数为0.369，价值为[629, 1950]，类别为红色。
  + 另一个子节点的样本量为12595，基尼系数为0.091，价值为[11994, 601]，类别为蓝色。
* 对于“blueTotalTowerDestroyed <= 6.5”节点：一个子节点的样本量为10890，基尼系数为0.017，价值为[91, 10799]，类别为红色。
  + 另一个子节点的样本量为1808，基尼系数为0.498，价值为[852, 956]，类别为红色。



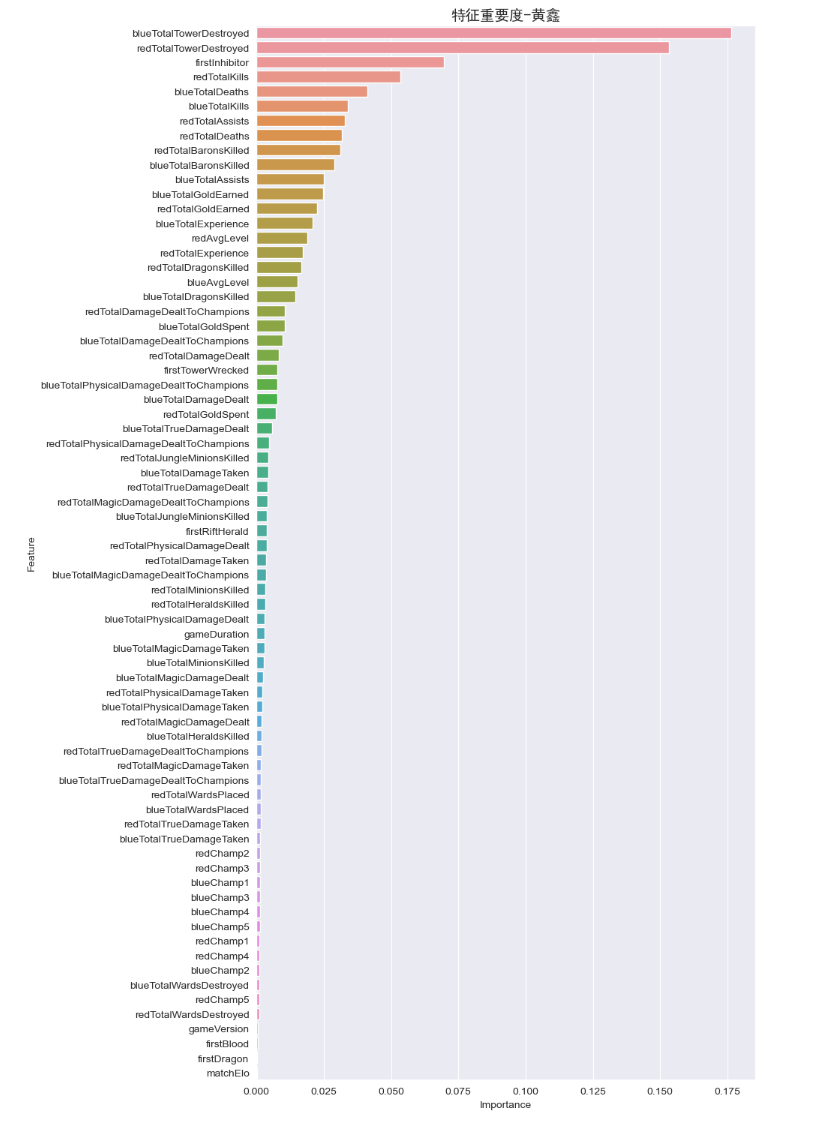
**杜金浩**

1. 数据预处理：数据集被加载并检查，没有发现缺失值。
   1. 目标变量'won'被分离为标签y，而其他列作为特征X。
   2. 使用LabelEncoder对非数值列进行了编码，将类别数据转换为数值数据。
   3. 使用StandardScaler对特征进行了标准化处理，这意味着特征被缩放和移位，使得它们具有均值为0和标准差为1的特性。
2. 数据集划分：数据集被划分为训练集和测试集，其中测试集大小为数据集的20%。
   1. 训练集包含27,872个样本，测试集包含6,968个样本。
3. SVM模型训练与评估：使用了一个线性核的支持向量机（SVM）分类器，并且一个非常小的正则化参数C值（0.000003）被用于模型训练。
   1. SVM模型在训练集上训练，并在测试集上进行预测。
   2. 模型的准确率为88.98%，这意味着模型能够以相当高的准确度预测游戏结果。
4. 混淆矩阵可视化：混淆矩阵显示了模型在预测蓝色队伍胜利和红色队伍胜利时的性能。
   1. 模型在预测红色队伍胜利方面表现较好（精度为0.99，召回率为0.83），但在预测蓝色队伍胜利方面表现稍差（精度为0.83，召回率为0.99）。
   2. 混淆矩阵中的数值显示，模型错误地将一些蓝色队伍胜利的样本预测为红色队伍胜利，这可能是由于数据不平衡或者模型复杂度不足导致的。



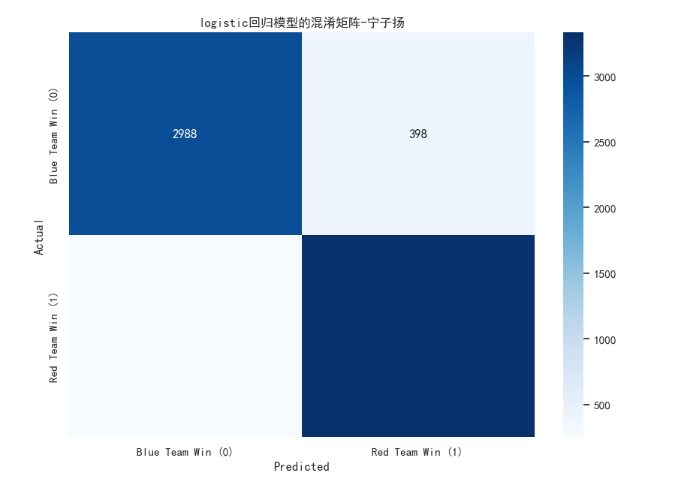
**黄鑫**

1. 数据预处理：数据集被加载并检查，没有发现缺失值。
   1. 特征'Unnamed: 0'和目标变量'won'被从数据集中移除。
   2. 使用了LabelEncoder对分类特征进行了编码，将类别数据转换为数值数据。
2. 数据集划分：数据集被划分为训练集和测试集，其中测试集大小为数据集的20%。
   1. 训练集包含6968个样本，测试集包含27872个样本。
3. RandomForest模型训练与评估：使用RandomForest分类器，并设置了100棵树。
   1. 模型在训练集上训练，并在测试集上进行预测。
   2. 模型的准确率为99.54%，这意味着模型能够以非常高的准确度预测游戏结果。
4. 特征重要性可视化：特征重要性通过计算每个特征在RandomForest模型中的重要性来确定。
   1. 重要性高的特征对模型的预测贡献更大。
   2. 特征重要性条形图显示了特征按照重要性从高到低排序。



**宁子扬**

1. 数据预处理：数据集被加载并进行了特征选择，只保留了与游戏结果相关的特征。
   1. 分类特征被转换为数值特征，使用map方法将’blue’和’red’映射为0和1。
   2. 目标变量'won'也被转换为数值特征，同样使用map方法。
2. 数据集划分：数据集被划分为训练集和测试集，其中测试集大小为数据集的20%。
   1. 训练集包含3386个样本，测试集包含3582个样本。
3. 特征标准化：使用StandardScaler对特征进行了标准化处理，这有助于防止某些特征因为数值范围较大而主导模型。
4. 逻辑回归模型训练与评估：使用逻辑回归模型对训练集进行了训练。
   1. 模型在测试集上进行预测，准确率为90.70%。
   2. 混淆矩阵显示，模型在预测蓝色队伍胜利（标签0）和红色队伍胜利（标签1）方面的表现是均衡的。
   3. 分类报告提供了精确度（precision）、召回率（recall）和F1分数，这些指标都显示了模型在两个类别上的表现是相似的。
5. 混淆矩阵可视化：使用Seaborn库的可视化混淆矩阵，展示了模型在预测各个类别时的性能。
   1. 混淆矩阵中的数值表示模型预测为红色队伍胜利（标签1）的实际是蓝色队伍胜利的样本数量，以及模型预测为蓝色队伍胜利的实际是红色队伍胜利的样本数量。



通过对比不同组员的数据预处理、模型训练、评估和可视化方法，我们可以得出以下结论：

1. 数据预处理：

- 所有同学都进行了数据预处理，包括加载数据、检查缺失值、选择相关特征、编码分类特征和标准化特征。

- 杜金浩和黄鑫使用了`LabelEncoder`对分类特征进行编码，而孙赫阳和宁子扬则使用了`map`方法。

- 宁子扬在特征选择阶段只保留了与游戏结果相关的特征，而其他同学没有明确提到这一点。

2. 模型训练：

- 杜金浩使用了SVM分类器，而黄鑫使用了RandomForest分类器。

- 孙赫阳使用了决策树可视化，而宁子扬使用了逻辑回归模型。

- 所有同学都使用了交叉验证（`train\_test\_split`）来划分数据集，并使用测试集来评估模型性能。

3. 评估指标：

- 所有同学都使用了准确率作为评估指标，而孙赫阳和宁子扬还使用了混淆矩阵和分类报告。

- 宁子扬使用了特征重要性可视化来展示哪些特征对模型的预测贡献最大。

4. 可视化：

- 杜金浩和宁子扬的可视化中包含了中文字体设置，这有助于更好地展示中文信息。

- 所有同学的可视化都是用来展示模型的性能，如混淆矩阵和特征重要性。

5. 模型性能：

- 模型的性能从88.98%到99.54%不等，这取决于所选的模型和参数。

- 所有模型都在预测游戏结果方面表现良好，但具体性能取决于所选的特征和模型。

6. 讨论：

- 特征选择对于模型的性能至关重要。宁子扬在特征选择阶段只保留了与游戏结果相关的特征，这可能有助于提高模型的性能。

- 不同的模型和参数可能会导致不同的性能。杜金浩使用了SVM分类器，而黄鑫使用了RandomForest分类器。这表明不同的模型和参数可能会导致不同的性能。

- 模型的泛化能力看起来不错，但高准确率可能掩盖了一些模型对数据细节的过度拟合。

- 为了进一步提高模型的性能，可以考虑增加特征、调整模型参数或尝试其他类型的分类器。

综上所述，不同的组员使用了不同的方法来进行数据预处理、模型训练、评估和可视化。这些方法的选择取决于所选的模型和参数。所有同学都展示了模型的性能，并提出了进一步改进的建议。

六．其他

上面五部分之外的内容，写在这里。

七．对本门课的感想、意见和建议

机器学习作为当今科技领域的前沿学科，已经在我们生活中的许多方面展现出其巨大的潜力和价值。在我们组的研究中，我们选择了Scikit-learn、TensorFlow和PyTorch这些强大的机器学习软件包，它们不仅提供了丰富的算法和模型，而且还具有良好的社区支持和文档资源，使得我们的研究工作得以顺利进行，并且也有利于我们更好的理解其工作原理。

我们组通过对英雄联盟电竞比赛数据的分析与分类，深刻体会到了机器学习在数据分析和模式识别中的强大能力。无论是决策树、逻辑回归，还是支持向量机和神经网络，每种模型都有其独特的优势和适用场景。这些模型的灵活性和多样性，让我们能够从不同的角度分析和理解数据，从而获得更加深入和全面的洞察。

在进行模型训练和评估的过程中，我们也认识到了机器学习在实际应用中的挑战。数据的预处理、特征工程和模型调参都需要大量的实验和经验积累。这让我们更加敬佩那些在机器学习领域不断探索和创新的科学家和工程师。

此外，机器学习的可视化工具也给我们留下了深刻的印象。通过可视化，我们不仅能够直观地看到模型的分类边界和决策过程，还能够发现数据中的潜在规律和异常点。这些可视化工具不仅增强了我们对模型的信任，还激发了我们对机器学习更深层次的兴趣。

对于机器学习这门科目，我们组建议更多的研究者能够参与到这个领域中来。机器学习不仅在电竞数据分析中有着广阔的应用前景，在医疗、金融、教育等众多领域都有着巨大的潜力。我们相信，随着技术的不断进步和应用的不断拓展，机器学习将会在未来的社会中扮演越来越重要的角色。

最后，我们组也希望通过我们的研究，能够为机器学习在电竞领域的应用提供一些新的思路和方法，同时也希望能够吸引更多的研究者加入到这个充满挑战和机遇的领域中来。