**2024美赛C题参考思路解析**

**Momentum in Tennis**

**题目：**

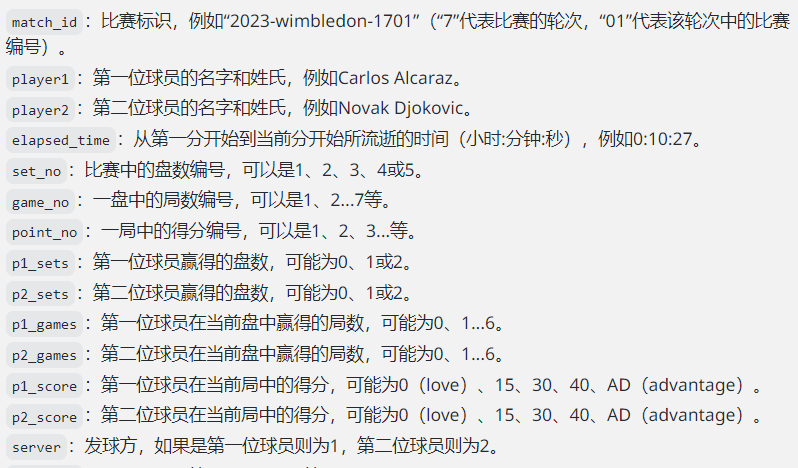
1. **开发一个模型**来捕捉比赛中的得分流动，并应用于一个或多个比赛，该模型应能标识哪位选手在比赛中表现更好，以及他们表现得有多好。此外，需要提供一个基于模型的可视化表示来描绘比赛流程。
2. **评估“动量”在比赛中的作用**。有观点认为比赛中的势头转换或成功的连续得分仅是随机发生的。使用你的模型/指标来评估这一观点。
3. **开发一个预测模型**，使用至少一场比赛的数据，预测比赛中势头的转换时刻。需要探讨哪些因素（如果有的话）与比赛中势头的改变最相关，并给出建议，如何根据过去的“动量”转换情况来指导选手准备新的比赛赛。
4. **测试模型**在一个或多个其他比赛中的表现。分析模型的预测效果如何，并讨论如果模型某些时候表现不佳不佳，可能需要考虑哪些因素在因素在未来模型中加以包含。同时，以包含。同时，评估模型对其他比赛（如估模型对其他比赛（如女子比赛对其他比赛（如女子比赛）、不同赛事、其他比赛（如女子比赛）、不同赛事、场地表比赛（如女子比赛）、不同赛事、场地表面，甚赛（如女子比赛）、不同赛事、场地表面，甚至其他运（如女子比赛）、不同赛事、场地表面，甚至其他运动（如乒乓球）的通子比赛）、不同赛事、场地表面，甚至其他运动（如乒乓球）的通用性。

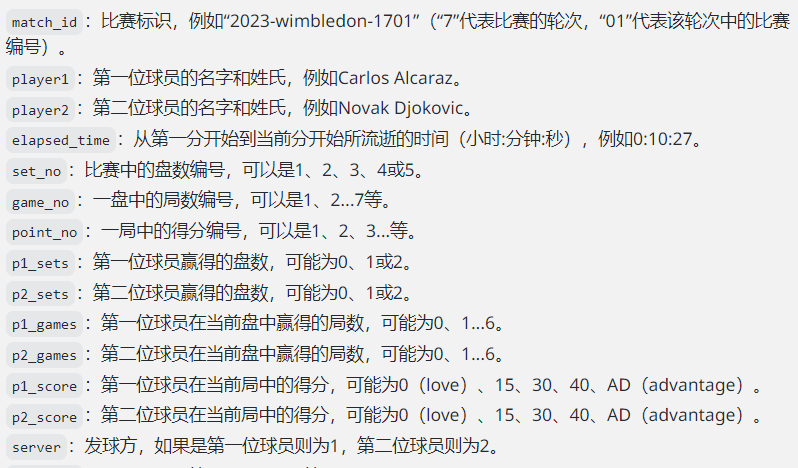
**解题流程：**

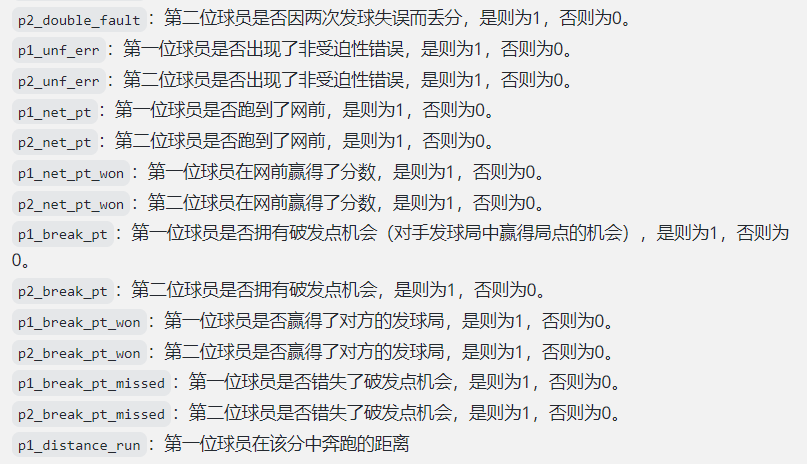
**第一步：收集和理解数据**

数据理解：首先，需要彻底理解提供的数据集Wimbledon\_featured\_matches.csv和data\_dictionary.csv。分析每个数据列代表的意义，特别是了解每一分的详情，包括比分、发球者、得分者以及其他可用的细节。

数据预处理：清理数据，处理缺失值和异常点。根据比赛ID (match\_id) 和赛集编号 (set\_no) 划分数据，确保能够针对特定比赛或赛集进行分析。







**第二步：分析每道问题**

针对问题一：

要开发一个模型捕捉比赛中随着每个得分发生的比赛流动，并应用于一个或多个比赛，我们需要先建立一个能够量化球员表现和比赛动态的系统。这一系统应综合考虑常规比分、发球优势、连续得分等因素。

首先，定义动量计算方法：动量是基于球员在比赛中连续赢得点数的概念。我们可以为每位球员设置一个动量得分，这个得分会根据他们赢得或失去点数而增减。比如，每赢得一分，动量加一；失分，则动量减少或重置。

接着，考虑发球优势：在网球比赛中，发球方通常有更高的得分概率。因此，在计算动量时，我们可以为发球方赢得的分数赋予较小的权重，以反映这一天然优势。

最后，加载数据并计算动量：首先，我们将加载比赛数据。然后，基于每一分的结果（由point\_winner列指示），我们更新两位球员的动量得分。如果球员是在他们的发球局赢得分数，我们可以稍微减少动量增加的量，以此反映发球优势。

可视化比赛流动：可以使用动量得分来可视化比赛流动。这将帮助我们直观地看到比赛中哪位球员处于领先位置，以及他们的领先程度。

代码：

这段代码首先初始化每位球员的动量得分为0。然后，它遍历每一分，根据赢球方更新动量得分，并考虑发球方的优势。最后，使用Matplotlib绘制动量得分，展示比赛的流动性。这个可视化帮助我们理解比赛过程中哪位球员处于领先位置以及他们的领先程度。

import pandas as pd

import matplotlib.pyplot as plt

# 加载数据

file\_path = '/mnt/data/2024-02-02-11-Z8ClxrQdXt8L3WnHdvcc.csv'

df = pd.read\_csv(file\_path)

# 初始化动量得分

df['p1\_momentum'] = 0

df['p2\_momentum'] = 0

p1\_momentum = 0

p2\_momentum = 0

# 遍历数据，计算动量

for i in range(len(df)):

point\_winner = df.loc[i, 'point\_winner']

server = df.loc[i, 'server']

if point\_winner == 1:

p1\_momentum += 1

p2\_momentum = 0

elif point\_winner == 2:

p2\_momentum += 1

p1\_momentum = 0

# 根据发球方调整动量

weight = 0.8 if server == point\_winner else 1.2

df.loc[i, 'p1\_momentum'] = p1\_momentum \* weight

df.loc[i, 'p2\_momentum'] = p2\_momentum \* weight

# 可视化

plt.figure(figsize=(10, 6))

plt.plot(df.index, df['p1\_momentum'], label='Player 1 Momentum')

plt.plot(df.index, df['p2\_momentum'], label='Player 2 Momentum')

plt.title('Match Momentum Over Time')

plt.xlabel('Point Index')

plt.ylabel('Momentum Score')

plt.legend()

plt.show()

针对第二问：三种方法

1. 统计分析

首先，我们可以通过统计分析来评估动量得分与比赛结果之间的关联度。如果动量确实对比赛结果有显著影响，我们应该能够看到高动量得分与获胜概率之间的正相关性。具体来说，我们可以：

计算相关性：使用皮尔逊相关系数或斯皮尔曼等级相关系数来评估球员动量得分与比赛结果之间的相关性。

进行回归分析：构建一个回归模型，以球员的动量得分为自变量，比赛结果（如获胜或失利）为因变量，来评估动量得分对比赛结果的影响力。

2. 随机性测试

对于教练提出的动量仅仅是随机波动的观点，我们可以通过以下方式进行测试：

随机模拟：对比赛进行随机模拟，其中点数的获胜完全是随机的，并计算在这种随机条件下的动量得分。重复模拟多次（例如1000次）以获取动量分布。

比较实际数据与随机模拟：将实际比赛数据中的动量得分分布与随机模拟的结果进行比较。如果实际比赛的动量得分显著偏离随机模拟的结果，这可能表明比赛中的动量不仅仅是随机波动。

3. 运行长度分析

另一种方法是分析连胜（或连败）的运行长度。如果比赛中的成功连续是随机的，那么我们期望连胜（或连败）的长度遵循特定的统计分布（如几何分布或二项分布）。

计算运行长度：对于每场比赛，计算连胜和连败的序列长度。

统计分布拟合：尝试将统计分布（如几何分布）拟合到运行长度数据上。如果实际数据与理论分布拟合良好，则支持教练的观点。

假设检验：进行假设检验（例如卡方检验）来确定运行长度的分布是否与随机过程相一致。

代码：

假设选择**统计分析**中的**计算相关性**方法来实现代码，这将帮助我们评估球员动量得分与比赛结果之间的相关性。可以使用皮尔逊相关系数，因为它可以衡量两个连续变量之间的线性关系强度。

假设我们已经有了一个DataFrame df，其中包含比赛数据，以及列p1\_momentum和p2\_momentum表示两位球员在比赛中每一分后的动量得分。此外，我们假设有一个match\_winner列，其中1表示球员1赢得比赛，2表示球员2赢得比赛。

import pandas as pd

import numpy as np

# 假设df是我们的DataFrame，包含比赛数据

# 获取比赛结束时的动量得分

final\_momentum = df.iloc[-1][['p1\_momentum', 'p2\_momentum']]

# 确定比赛胜者

match\_winner = df.iloc[-1]['match\_winner']

# 将比赛胜者转换为与动量得分相关的分数

# 假设如果球员1赢，则赋值1，球员2赢，则赋值-1，这样可以反映出胜者的动量应该更高

winner\_momentum\_score = 1 if match\_winner == 1 else -1

# 计算皮尔逊相关系数

# 注意，这里我们使用的是最终动量得分与一个标量（比赛胜者的动量分数），这不是典型的用法，

# 通常我们会计算两个变量的相关性。因此，这个计算将简化为分析最终动量得分与比赛胜者之间的关系

correlation\_p1 = np.corrcoef(final\_momentum['p1\_momentum'], winner\_momentum\_score)[0, 1]

correlation\_p2 = np.corrcoef(final\_momentum['p2\_momentum'], winner\_momentum\_score)[0, 1]

print(f"Player 1 Momentum to Match Outcome Correlation: {correlation\_p1}")

print(f"Player 2 Momentum to Match Outcome Correlation: {correlation\_p2}")

针对问题三：

要解决这个问题，我们需要开发一个模型，该模型能够预测比赛中动态变化的时刻，即比赛的流向何时从一名球员转向另一名球员。这涉及到识别可能预示着比赛动向改变的指标。此外，我们还需要考虑如何使用过去比赛中的“动量”转换数据来指导球员面对新的对手。

为了开发这个模型，我们需要分析至少一场比赛的数据，找出与比赛动向转变相关的因素。这些因素可能包括，但不限于，球员的体能状态、比赛中的失误次数、发球成功率、赢球点和破发点的实现率等。

可以考虑使用逻辑回归、随机森林或支持向量机等分类算法，这些算法能够处理二分类问题（即比赛动态是否即将改变）。

代码：

假设使用逻辑回归模型，因为它适用于处理二分类问题（即预测比赛动态是否即将改变）。

import pandas as pd

import numpy as np

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix

# 这里使用pd.read\_csv()或任何适合的方法加载你的数据集

df = pd.read\_csv('your\_dataset.csv')

# 假设我们基于发球成功率、破发点转换率等创建特征

# 这里我们简化，只使用两个特征作为示例

features = df[['service\_success\_rate', 'break\_point\_conversion\_rate']]

target = df['momentum\_shift'] # 假设已经有一列标记动量是否改变，1表示是，0表示否

#模型训练和评估

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, target, test\_size=0.3, random\_state=42)

model = LogisticRegression()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print(f"Accuracy: {accuracy}")

print(f"Confusion Matrix: \n{conf\_matrix}")

应用“动量”数据给出建议

基于历史比赛中“动量”变化的分析，我们可以给球员提供关于如何准备和应对不同对手的建议：

a. 分析动量变化模式

识别模式：分析球员过去比赛中动量的转换点，识别是否存在特定的模式，例如在特定比分时更容易发生动态转变。

对比对手：分析即将对阵的对手的类似数据，看看是否有相似的动量转换模式或者对手在某些方面的脆弱性。

b. 制定策略

强化训练：根据分析结果，强化球员在可能的转折点上的表现，比如提高关键分的发球质量或者加强心理准备。

调整比赛策略：根据对手的脆弱点和自己的强项，调整比赛策略，比如在对手体能下降或心理不稳定时加大进攻力度。

针对问题四：选取新的数据集进行验证。