Assumption

:已经写在论文里了

Model:

performance metric

建模思路：performance只是衡量当前玩家在**比赛时的表现**，评估这个表现重点在于玩家在比赛中如何打出漂亮的球 是否上网 是否错失机会 第一个发球是否发过网...

就好像跳水表演 你在欣赏一个跳水运动员 这不需要比赛的胜负和对手的实力决定

所以：

在建模中使用的数据：

```
player_1_perf_col=
['p1_ace','p1_winner','p1_double_fault','p1_unf_err','p1_net_pt',
'p1_net_pt_won','p1_break_pt','p1_break_pt_won','p1_break_pt_missed','p1_distance
_run']
player_2_perf_col=
['p2_ace','p2_winner','p2_double_fault','p2_unf_err','p2_net_pt',
'p2_net_pt_won','p2_break_pt','p2_break_pt_won','p2_break_pt_missed','p2_distance
_run']
```

思路：利用玩家在赛场上表现的评价性指标，加权后给玩家打分

加权参数：逻辑回归调节

因为在假设中 这些量都是相互独立的（理解：比如这个玩家打出一个相对糟糕的球和其后期打出制胜球之间没有太多关系）也是二元变量 采用逻辑回归

方法：共31场比赛 遍历比赛集，每次拿1组比赛做测试集 剩下的30场比赛做训练集，保存的final_performance.csv为每个比赛用其余三十场比赛的预测结果

逻辑回归得到胜利的概率

表现得分计算方式：x=log(单场逻辑回归probability*10*(2381/(2381+4903))+1) if 自己发球，else x=log(单场逻辑回归probability*10*(4903/(2381+4903))+1) if 对手发球

log是因为分数两极化有的比较严重想削一削，+1是保证正数， 2381/（2381+4903）是消除发球对performance的作用 2381是全场比赛中接发赢的，4903是发球赢的

current prformance 是由近3局的每局performance求平均

图在代码 `get_csv.ipynb` 里面画了，不好看建议用别的package重新画下。挑最后一组(题目里说的那场比赛) 再随便挑一两组（at least one）可视化一下（legend要改）

momentum metric

由3个指标：`current performance`，`mark`，`win_factor` 加权得到 后面两个指标怎么生成的问鹏

加权过程：（以`current_performance`为例 其余两个都经过了相同的黑箱处理）

momentum

1. 初始`current_performance = np.mean(current_performance)`所有选手都一样
2. 拿第一个`current_performance`作为第一个momentum[performance]的值
3. 对第二个momentum[performance]，用初始current_performance和momentum[performance]高斯加权共同预测 把加权的值作为第二行的momentum

```
def get_weight(length_):
    temp=[]
    for i in range(length_):
        temp.append(gaussian_distribution_value_at_x(i*0.5))
    temp=np.sort(np.array(temp))
    return normalize(temp)
```

高斯加权函数 输入长度输出权重 越新的权重越大

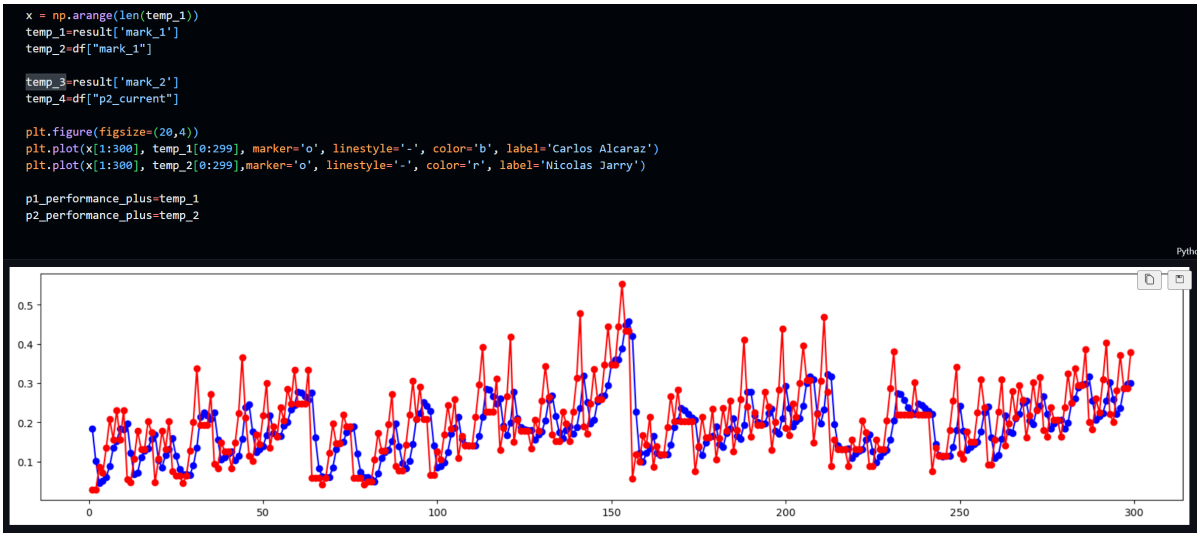
4. 以此类推 直到一个game 结束
5. 下一个game, 拿上一局最后一个momentum作为下一局的开端 因为是新局 而且中场休息momentum要削弱 向均值靠拢

```
def mom_reg_adj(mom,mean_0_1):
    d=np.sign(mom-mean_0_1)
    return mean_0_1+d*(1-np.e**(-(np.abs(mom-mean_0_1))))
    每局结束后 momemtum都要经过一次这个函数mean_0_1是当前量 （performance）的全体均值
```

以此类推 直到一个match结束 代码：momentum.ipynb

最后的momemtum三个指标怎么加权问鹏

黑盒的实际作用：类似于平滑（有一格的滞后）



假设检验

代码：problem_2.ipynb

思想：生成随机数组【0，1】，长度100，000

把数组的长度归成分布 如【0，0，1，1，1，1，0，0，1】-->[2,4,2,1]

momentum 把指标正数变成1，负数变成0，

验证这两个分布是否具有同分布--> Anderson-Darling检验

-->全部的momentum都是99%的概率认为不是随机的

zip 数据说明：

- player_12_2.csv: get_csv里面要用的整理过的原始数据
- get_csv.ipynb: 获得performance
- problem2.ipynb:假设检验
- performance_final 这里面的？行是用来分隔每match的，选手表现的最终数据
- momentum_calculate.py 计算momentum 加权
- momemtum.ipynb, momentum黑盒子
- mark_plus.csv: momentum的另外两个指标
- final_momentum_data.csv：最终的momentum数据
- data_process ？ 数据处理