数据挖掘第3次大作业--可解释的分类模型

1. 任务简介

1.1 课题背景

四子棋是一种广泛流传的连棋类益智游戏，由Howard Wexler在1974年推出[1]。四子棋的规则非常简单，在标准的棋局中，棋盘共有7列6行，直立放置。两个玩家各执一种颜色的棋子参与游戏，游戏过程中，玩家轮流将自己的棋子投入某一列，该棋子会在重力作用下落到该列最低的空行中，直到某个玩家有4枚棋子以纵、横、斜方向连成一线时获胜。若棋盘下满时还未决出胜负则为平局。

标准7x6棋盘的四子棋解法已分别被Victor Allis及James D. Allen破解，在完美玩法中，先手方只要把第一枚棋子放在最中间一行，便保证不败。后手方在最中间的左或右一行下第一枚棋子，便可达成和局。在左或右最边一行下第一只棋子，亦有可能可取得胜利[2]。

在本课题中，我们将设计一系列的分类模型，依据棋盘在游戏进行中的状态，预测该棋局最终的胜负情况。该问题在数学上已有完备的解决方法，因此我们关注的重点在于如何使计算机模型自动学习到棋局的特征，在预测中达到较高的准确率，同时我们也将关注模型的可解释性，观察和分析模型是否能学习到有意义的规则，并与现有的数学推导规则进行比较。

1.2 任务挑战

虽然该问题对应的数学模型不是非常复杂，且有较多的研究，但在不加入任何数学知识的指导时，分类器能否、以及如何自动学习四子棋的规则，从而做出准确的判断，依然是一个值得探究的问题。总的来说，任务的挑战来自于如下几个方面：

1. 较高的预测准确率：本问题中给出的是一个静态棋盘，没有动态下棋的过程，要从此阶段推测最终棋局情况，存在一定的难度，且现有的下棋规则也无法简单应用在其中。
2. 模型可解释性的探究：关于模型的可解释性，Kim等人提出了一个通用公式[5]：

$$Argmax_E Q(E|Model, Human, Data, Task)$$

其中，Q是解释性的评价方法，E是实现可解释性的具体算法，因此可解释性的过程就是寻求特定任务（Task）的解释，使得特定的人群（Human）对于特定的数据（Data）和模型（Model）获得最大程度的理解。

在传统的模型可解释性相关工作中，来自于人机交互领域的方法评价通常来自于模型学习的规则与人工标注之间的差别，或是来自用户实验中探究在实际应用时用户对于模型解释的接受程度[4]，从而直接对模型做出解释的质量进行评价。在模型为主的评价方法中，则通常通过特征的重要性、模型结果可视化等方式。

由于本问题中没有Human这一主体的加入，我们在模型可解释评价中沿用模型为主的评价方法，主要从特征和模型结果的角度，评价模型的可解释性。

3. 模型规则的实际意义：四子棋是一个在数学上已完全解决的、规范化的实际问题，所以从解释性分析中，提取指导棋手的策略也非常重要。

2. 数据集及其预处理

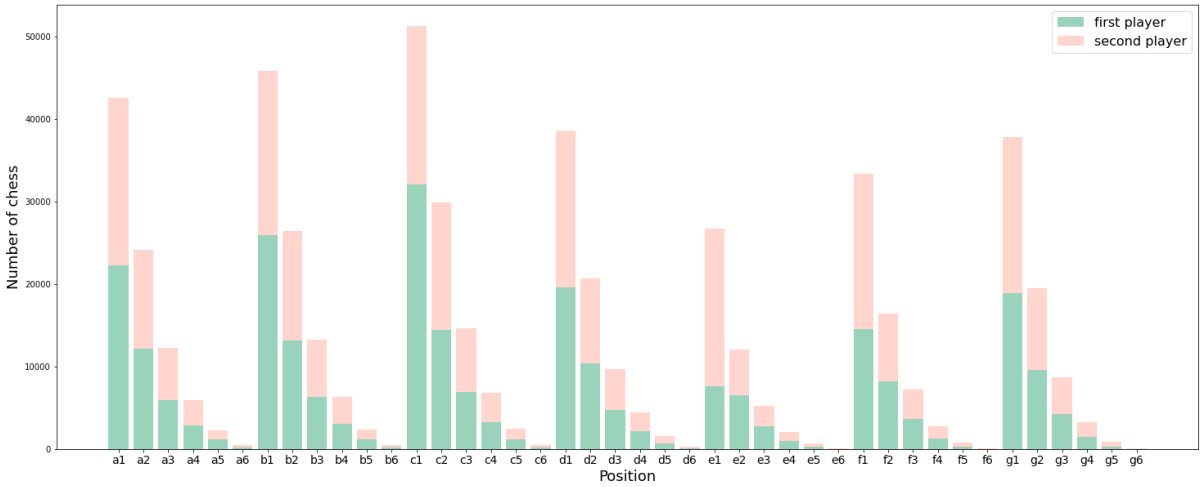
2.1 数据集描述

本课题中使用UCI机器学习数据集库中的connect-4数据集[3]。该数据集对应的是7*6的标准棋盘四子棋游戏，包含67557个已落下8个棋子的棋盘状态实例，和两个玩家在此状态下均采取最优策略之后对应的博弈结果，无缺失值。

这67557个实例中，标签分布情况如下：

Label	Count	Percentage
win	44473	65.8%
loss	16635	24.6%
draw	6449	9.6%

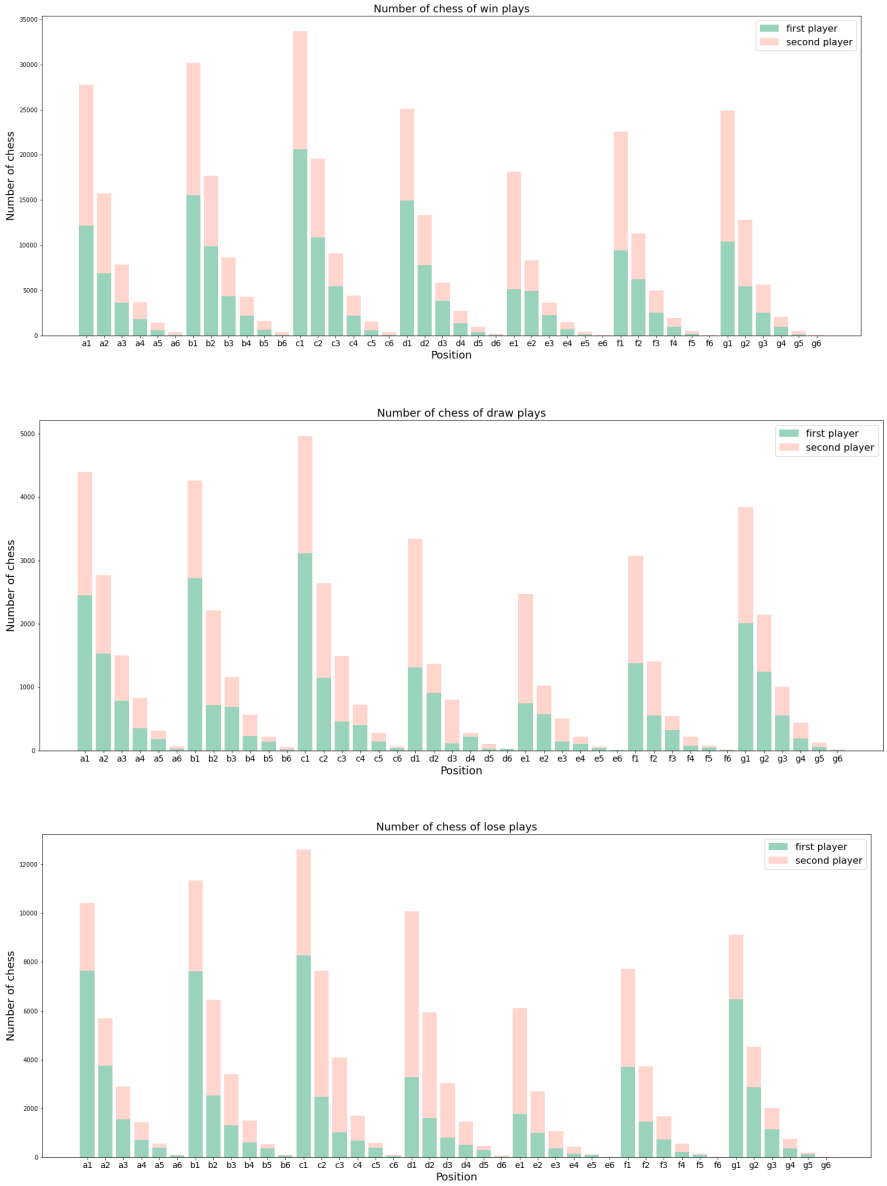
每个位置落子的数量统计如下：



该图中，横坐标表示42个格点，纵坐标表示数据集中落子总数，绿色为第一个玩家的落子数量，粉色为第二个玩家的落子数量。从图中可以看出，每一列的落子数量都随着行数升高而增大，到最后一行时几乎没有落子，这是由四子棋的游戏规则决定的，四子棋的棋子只能在每一列从下往上堆叠放置。

关注不同的列之间的变化，可以发现整体而言，两个玩家都更喜欢在前三列落子，在第5、6列落子数量均较少。

以下对赢、平局和输三种情况，分别统计每个位置落子数量：



从统计结果中可以得到几点简单的观察，例如，赢、平局、输三种情况下，第二个玩家在a1处落子的比例逐渐降低，而在d这列落子的比例逐渐增加。但是，如果从单个样本进行观察，在没有四子棋的专业知识的情况下，很难人工判断该棋局接下来会如何发展，因此，有必要借助机器学习模型构建分类器，解决问题。

2.2 数据集预处理

考虑到数据集给定的特征是离散值，无法作为多层感知机等模型的输入，所以我们将所有的属性替换为数值型属性，具体来说，我们用1表示'x'(x玩家落子的位置)，-1表示'o'(o玩家落

子的位置)，0表示'b'(双方都没有落子的位置)。

另外，针对类别标签，我们使用1表示'win'(先落子玩家获胜)，-1表示'loss'(后落子玩家获胜)，0表示'draw'(平局)。

3. 模型选择

3.1 可解释模型

随着机器学习，尤其是深度学习的发展，模型的规模越来越大、性能越来越强，但许多模型只能作为一个“黑盒子”，即使用者只能观测其输入、输出，而无法从参数中得到模型做出判断的原因解释。然而在机器学习的广泛应用中，其他专业的从业人员更希望得到模型的解释性结果，因此近年来，模型的可解释性逐渐受到重视，也有越来越多的工作专注于模型的可解释性研究。考虑到本数据集的特征数量较少、特征取值较单一，不适用于复杂的模型，因此选择如下几个模型进行可解释性的分类研究：

3.1.1 线性回归模型

本问题是一个三分类的问题，因此采用线性+softmax回归的模型结构进行预测，该模型的可解释性体现在，各特征通过线性组合的方式对预测结果产生影响，因此可以通过比较特征的权重，解释各特征对预测的重要性。在本工作中，该模型通过自主代码实现和调用sklearn库两种方式进行了实现。

softmax回归是对用于二分类的逻辑回归（Logistic Regression）模型的推广，在逻辑回归中，样本X与模型所学习的权重w相乘后通过sigmoid函数，获得样本X属于两个类别的概率：

$$P(Y=1|x)=\frac{\exp(w^Tx+b)}{1+\exp(w^Tx+b)}$$
$$P(Y=0|x)=\frac{1}{1+\exp(w^Tx+b)}$$

在此模型中，仅需要对sigmoid函数进行扩展，对x的线性组合使用softmax函数运算即可得到多分类模型中，样本属于每一类的概率：

$$h_\theta(x) = \begin{bmatrix} P(y=1|x;\theta) \\ P(y=2|x;\theta) \\ \vdots \\ P(y=K|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\theta_j^T x)} \begin{bmatrix} \exp(\theta_1^T x) \\ \exp(\theta_2^T x) \\ \vdots \\ \exp(\theta_K^T x) \end{bmatrix}$$

模型的参数为K*M的θ矩阵，K为分类数量，M为特征维度（X的维数），使用批量梯度下降的方法进行训练。具体而言，h产生数据集D={(x1,y1),(x2,y2), ..., (xn, yn)}的似然为

$$\begin{aligned} L(x,y,\theta) &= \prod_{i=1}^{n} P(x_i)\vec{h}_\theta(x_i)[y_i] \\ &\propto \prod_{i=1}^{n} \vec{h}_\theta(x_i)[y_i] \end{aligned}$$

其中，[y]表示从向量中得到第y维的数据。对数似然为

$$\begin{aligned} l(x,y,\theta) &\propto \sum_{i=1}^{n} log(\vec{h}_\theta(x_i)[y_i]) \\ &\propto \sum_{i=1}^{n} (\theta_{y_i}^T x_i - ln(\sum_{j=1}^{K} exp(\theta_j^T x_i))) \end{aligned}$$

因此，对应的损失函数可写作

$$\begin{aligned} Loss &= -\frac{1}{n}\sum_{i=1}^{n} log(\vec{h}_\theta(x_i)[y_i]) \\ &:= -\frac{1}{n}\sum_{i=1}^{n} log(p_{x_i,y_i}) \end{aligned}$$

对每个参数求导

$$\begin{aligned} \frac{\partial Loss}{\partial \theta_{j,m}} &= -\frac{1}{n}\sum_{i=1}^{n} \frac{1}{p_{x_i,y_i}} \frac{\partial p_{x_i,y_i}}{\partial \theta_{j,m}} \\ &= \frac{1}{n}\sum_{i=1}^{n} \frac{1}{p_{x_i,y_i}} p_{x_i,y_i} * (p_{x_i,j} * x_{i,m} - x_{i,m}\mathbb{I}(y_i = j)) \\ &= \frac{1}{n}\sum_{i=1}^{n} (p_{x_i,j} * x_{i,m} - x_{i,m}\mathbb{I}(y_i = j)) \end{aligned}$$

根据该求导，更新参数取值

$$\theta_{j,m} = \theta_{j,m} - \alpha \frac{\partial Loss}{\partial \theta_{j,m}}$$

其中，α为学习速率超参数，在训练时指定。

参数更新过程会一直进行，直到连续几次训练中，损失函数值不再下降，或达到某指定的最大迭代次数max_iter为止。

逻辑回归方法虽然原理上非常简单，但运算速度快、具有较强的解释性，在线性可分问题上具有良好的表现，因此在本问题中，我们首先就想到了采用此方法进行可解释模型的构建。

3.1.2 CART决策树模型

决策树是一种常用的分类器，采用树形结构对监督数据进行分类，具有较强的可解释性，根据划分节点选择特征的方法不同，决策树有多种实现方式。在本工作中，因为数据的每个属性都是离散的、有三种可能取值的特征，因此我们直接使用了基尼系数作为特征选择依据的CART决策树进行实验。

具体而言，对于一个数据集，自顶向下构建一棵决策树。开始时，根节点包括所有的样本，计算此时根节点的Gini不纯度

$$Gini(A) = 1 - \sum_{i=1}^{C} p_i^2$$

其中，pi表示属于第i类的概率，计算中用该节点的样本中属于第i类的比例代替。接着，按照每一个属性的每一种可能的划分方式将数据集划分为两个部分，计算划分后的两个子节点的Gini不纯度的加权和为Gini指数

$$Gini(p) = Gini(ch_1)\frac{n_1}{n_1+n_2} + Gini(ch_2)\frac{n_2}{n_1+n_2}$$

其中，p为当前的划分方式，ch1和ch2代表两个子节点，n1和n2是两个子节点包含的样本个数。因此，划分后的不纯度减少量为

$$\Delta Gini(A,p) = Gini(A) - Gini(p)$$

选择使Gini不纯度较少最多的属性和划分点，作为该节点的划分方式。

以此类推，不断对节点进行划分，直到某个节点下所有的样本都属于同一类（Gini不纯度为0）为止。

在应用实现中，可通过控制决策树的深度、一个节点至多考虑的属性数量或叶节点至少包含的样本数量等方式，对决策树进行剪枝。

由于本问题的所有属性都是离散取值，树结构的模型应能取得较好的效果。从可解释性的角度考虑，决策树通过层次推理的方式实现最终的分类，能够从数据中总结出if-then-else的规则，具有较强的可解释性。

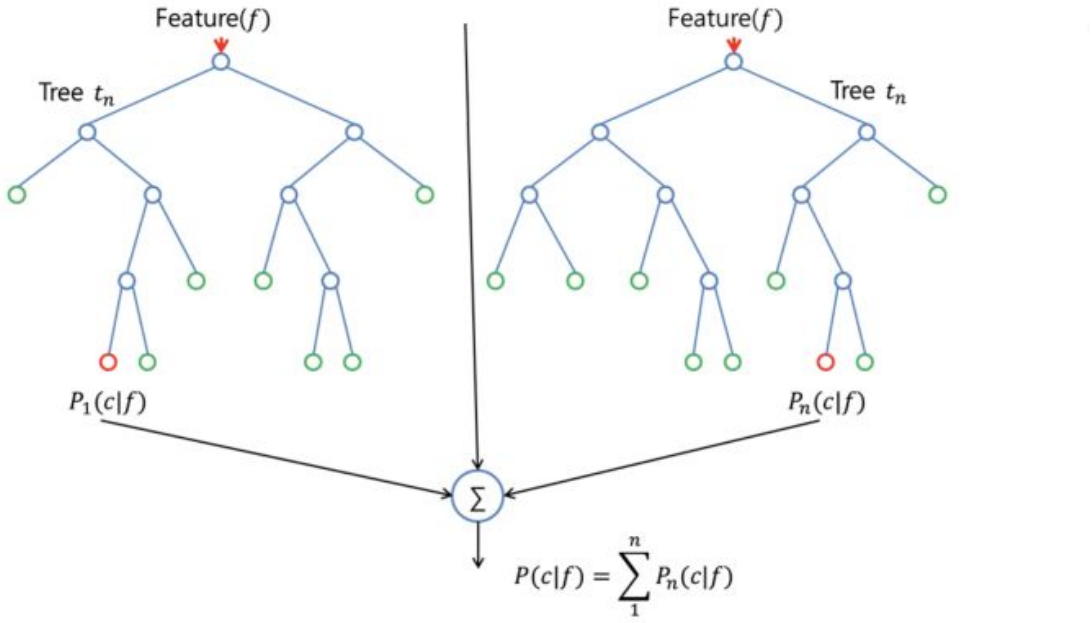
3.2 不可解释模型

3.2.1 随机森林模型

集成学习（ensemble）思想是为了解决单个模型或者某一组参数的模型所固有的缺陷，从而整合起更多的模型，取长补短，避免局限性。随机森林就是集成学习思想下的产物，将许多棵决策树整合成森林，并合起来用来预测最终结果。

随机森林模型的关键在于使用了bagging方法。Bagging基于自助采样法。给定包含m个样本的数据集，先随机取出一个样本放入采样集中，再把该样本放回初始数据集，使得下次采样时该样本仍有可能被选中(有放回的采样)。这样，经过m次随机采样操作，得到含m个样本的采样集，初始训练集中有的样本再采样集里多次出现，有的则从未出现。采样出T个含有m个训练样本的采样集，然后基于每个采样集训练出一个基学习器，再将这些基学习器进行结合。

而随机森林实际上是一种特殊的bagging方法，它将决策树用作bagging中的模型。首先，用上述方法生成m个训练集，然后，对于每个训练集，构造一颗决策树，在节点找特征进行分裂的时候，并不是对所有特征找到能使得指标（如信息增益）最大的，而是在特征中随机抽取一部分特征，在抽到的特征中间找到最优解，应用于节点，进行分裂，算法示例图如下：



随机森林的方法由于有了bagging，也就是集成的思想在，实际上相当于对于样本和特征都进行了采样（如果把训练数据看成矩阵，就像实际中常见的那样，那么就是一个行和列都进行采样的过程），所以可以避免过拟合。

3.2.2 梯度提升树模型

梯度提升树（GBDT）模型采用了Boosting的集成学习方法，由多棵决策树组成，所有树的结论累加起来做最终答案，但是却和传统的Adaboost有很大的不同。Adaboost是利用前一轮迭代弱学习器的误差率来更新训练集的权重，这样一轮轮的迭代下去。而在GBDT的迭代中，假设我们前一轮迭代得到的强学习器是$ft\text{-}1(x)$, 损失函数是$L(y, ft-1(x))$, 本轮迭代的目标是找到一个回归树模型的弱学习器$ht(x)$，让本轮的损失函数$L(y, ft(x))=L(y, ft-1(x)+ht(x))$最小。也就是说，本轮迭代找到决策树，要让样本的损失尽量变得更小。

但是这个损失的拟合不好度量，针对这个问题，Freidman提出了用损失函数的负梯度来拟合本轮损失的近似值，进而拟合一个回归树。第t轮的第i个样本的损失函数的负梯度表示为

$$r_{ti} = -\left[\frac{\partial L(y_i, f(x_i)))}{\partial f(x_i)}\right]_{f(x)=f_{t-1}(x)}$$

利用数据集我们可以拟合一颗回归树，得到了第t颗回归树，其对应的叶节点区域$Rtj, j=1,2,...,J$。其中J为叶子节点的个数。针对每一个叶子节点里的样本，我们求出使损失函数最小，也就是拟合叶子节点最好的的输出值如下：

$$c_{tj} = \underbrace{\arg\min}_{c} \sum_{x_i \in R_{tj}} L(y_i, f_{t-1}(x_i) + c)$$

这样我们就得到了本轮的决策树拟合函数如下

$$h_t(x) = \sum_{j=1}^{J} c_{tj} I(x \in R_{tj})$$

从而本轮最终得到的强学习器的表达式如下：

$$f_t(x) = f_{t-1}(x) + \sum_{j=1}^{J} c_{tj} I(x \in R_{tj})$$

我们通过其损失函数的负梯度的拟合，就可以用GBDT来解决我们的分类回归问题。假设

类别数为K，则此时我们的对数似然损失函数为：

$$L(y, f(x)) = -\sum_{k=1}^{K} y_k \log p_k(x)$$

其中如果样本输出类别为k，则yk=1。第k类的概率$pk(x)$的表达式为：

$$p_k(x) = \exp(f_k(x)) / \sum_{l=1}^{K} \exp(f_l(x))$$

集合上两式，我们可以计算出第t轮的第i个样本对应类别l的负梯度误差为

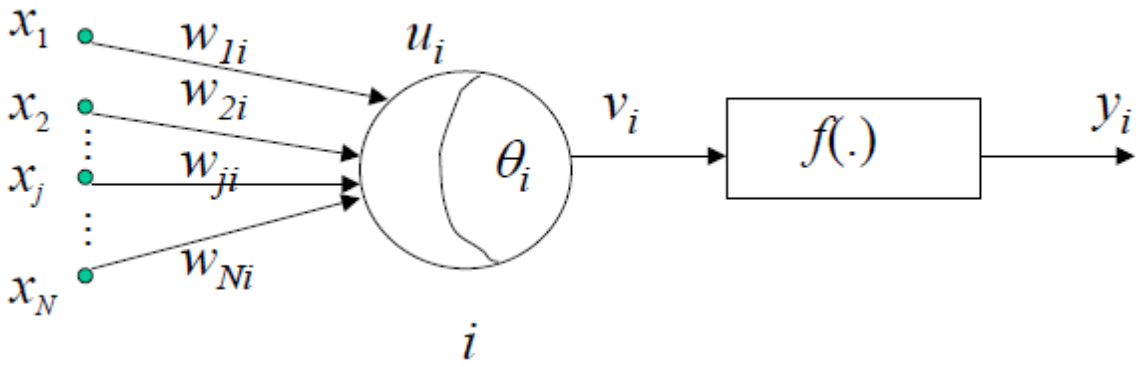
$$r_{til} = -\left[\frac{\partial L(y_i, f(x_i)))}{\partial f(x_i)}\right]_{f_k(x)=fl,t-1(x)} = y_{il} - p_{l,t-1}(x_i)$$

对于生成的决策树，我们各个叶子节点的最佳负梯度拟合值为

$$c_{tjl} = \underbrace{\arg\min}_{c_{jl}} \sum_{i=0}^{m} \sum_{k=1}^{K} L\left(y_k, f_{t-1,l}(x) + \sum_{j=0}^{J} c_{jl} I(x_i \in R_{tjl})\right)$$

3.2.3 多层感知机模型

多层感知器（Multilayer Perceptron，MLP）是一种前向结构的人工神经网络，映射一组输入向量到一组输出向量。主要优势在于其快速解决复杂问题的能力。MLP可以被看作是一个有向图，由多个的节点层所组成，每一层都全连接到下一层。具体来说，多层感知机的基本结构由三层组成：第一输入层，中间隐藏层和最后输出层，输入元素和权重的乘积被馈给具有神经元偏差的求和结点。除了输入节点，每个节点都是一个带有非线性激活函数的神经元（或称处理单元），单个节点结构如下：



若每个神经元的激活函数都是线性函数，那么，任意层数的MLP都可被约简成一个等价的单层感知器。正是由于激活函数的存在，使得MLP成为了感知器的推广，克服了感知器不能对线性不可分数据进行识别的弱点。

我们通常使用反向传播算法的监督学习方法来训练MLP。多层感知器遵循人类神经系统原理，学习并进行数据预测。它首先学习，然后使用权重存储数据，并使用算法来调整权重并减少训练过程中的偏差，即实际值和预测值之间的误差。

具体来说，我们可以将输出节点 j 的第 n 个数据点的误差表示为

$$e_j(n) = d_j(n) - y_j(n)$$

其中d是目标值，y是由感知器预测的值。调整节点权重的方式是，尝试通过修正节点权重最小化输出的整体误差

$$\mathcal{E}(n) = \frac{1}{2}\sum_j e_j^2(n)$$

使用梯度下降，每个权重的修正量为

$$\Delta w_{ji}(n) = -\eta\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} y_i(n)$$

其中yi是前一个神经元的输出，η是学习率。值得注意的是，由于上述式子中包含求导的过程，所以为了使用反向传播算法进行有效学习，激活函数必须限制为可微函数。

4. 实验及结果分析

4.1 实验设计

4.1.1 数据集划分

我们将数据集随机划分为5份，从而可以使用5折交叉验证得出各模型的效果。具体来说，我们根据原数据集中各条数据集的编号除以5的余数（即0,1,2,3,4）将数据集划分为5个部分，前两部分分别包含13512条数据，后三部分分别包含13511条数据。

4.1.2 参数设置

在线性回归算法中，在我们自主实现的算法中，我们设置初始学习率为0.05，学习率下降率为0.8，early stop patience为20步迭代，至多迭代3000次。在sklearn库调用中，我们设置初始学习率为0.02，至多迭代3000次。

在决策树算法中，我们不限制树的最大深度，使用最佳划分方式对每个节点进行划分,每个非叶子结点至少需被划分为2个子节点。

在随机森林算法中，我们共构造100颗决策树，规定每棵决策树的最大深度为100，每个非叶子结点至少需被划分为2个子节点。

在梯度提升树算法中，我们共构造3000棵决策树，规定每棵决策树的最大深度为6，每个非叶子结点至少需被划分为2个子节点。

在多层感知机算法中，我们使用一层隐藏层，该层的隐藏单元数为200，激活函数选择为tanh函数，优化方法我们使用adam算法，学习率设置为0.001，正则系数设置为0.00001，最大迭代轮数设置为500。此外，为了防止过拟合，每次训练的过程中，我们随机选择10%的训练数据作为验证集，并在验证集效果不变好时采用early stopping策略。

4.1.3 评价指标选择

我们使用Macro-F1来评价各个模型的多分类效果。具体来说，针对每个类别（即win, draw, loss），我们计算其TP（true positive，预测是正确的正样本），FP（false positive，预测是错误的正样本），TN（true negative，预测是正确的负样本），FP（false positive，预测是错误的负样本）。

在此基础上，我们计算各个类别的Precision，Recall和F1-score：

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\text{-}score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

然后，我们将上述三类的F1-score取平均值，即为我们所求的Macro-F1。

值得注意的是，我们使用了5折交叉验证来评价模型的效果，故将5次测试集上Macro-F1的平均值作为最终的评价指标，并通过各个模型五次结果的方差来衡量模型性能的稳定性。

4.2 分类结果及分析

我们使用的5种模型多分类效果及稳定性如下表所示（其中最好的结果我们用加粗注明，较差的结果我们用下划线注明）：

模型	Macro-F1	方差
线性回归（自主实现）	0.4929	5.732*10-6
线性回归（使用sklearn库）	0.4968	**5.356*10-6**
CART决策树	0.6185	2.631*10-5
随机森林	0.6409	1.408*10-5
梯度提升树	**0.7326**	1.422*10-5
多层感知机	0.6955	9.293*10-5

根据以上结果，我们可以发现，我们自主实现的线性回归模型和调用sklearn库实现的结果基本一致，说明了我们手动实现结果的正确性，而线性回归模型由于其本身能力较弱，无法拟合特征与特征之间的非线性关系，所以在所有的模型之中分类效果最差。

而观察三种树模型的效果，我们可以发现，仅仅使用一棵决策树，虽然效果优于线性回归模型，但是和使用集成学习融合多个决策树模型的结果相比要更差一些，而相比于使用简单bagging方法的随机森林，使用更先进的梯度提升方法的梯度提升树模型效果更好，该模型也在所有的方法中取得了最优的性能。

而多层感知机模型由于具有强大的拟合特征之间非线性关系的能力，所以在所有的模型中，分类效果仅次于梯度提升树模型。我们认为，对于棋类游戏的形势判断，基于规则的方法相比于函数拟合的方法更加有效，这点由决策树模型效果优于线性回归模型这一现象也可以得到验证。

而在模型效果的稳定性方面，由于线性模型较简单，故模型稳定性最好，但是由于其较差的分类性能，使得这点意义不大，而多层感知机模型的稳定性最差，这也反映了在该问题中，通过对特征的非线性关系进行拟合具有较大的随机性，而基于规则的树模型效果会更加稳定，是更好的解决方案。

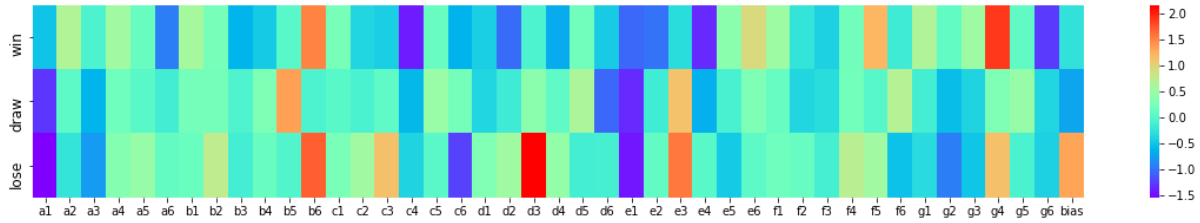
5. 可解释模型评价与规则提取

可解释性是本工作的一个重点，因此除了分类结果以外，我们还对两个可解释模型的结果进行了详细的分析。以下我们首先直接观测模型所学习的特征权重，再利用SHAP方法[7]对模型的输出进行解释，最终结合两者，提出了量化评价模型可解释性的指标。

5.1 模型参数可视化

5.1.1 线性回归模型

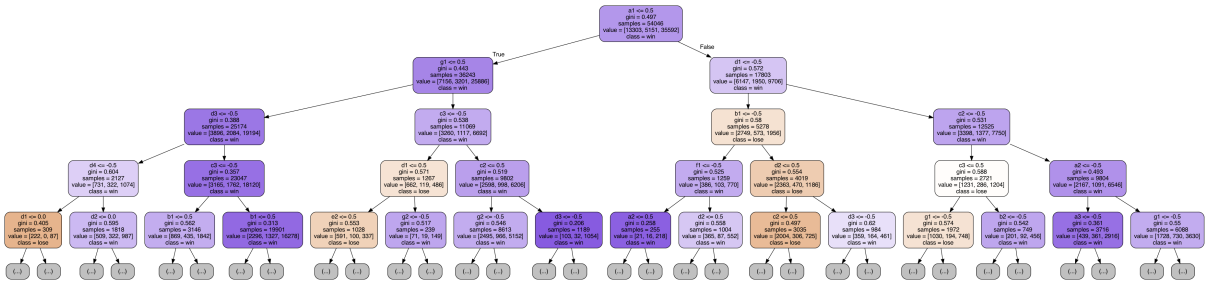
线性回归模型中，将每个特征对应的权重和bias绘制成热图如下所示。



其中，横坐标为棋盘每个位置对应特征，纵轴为三分类的每种情况，每个格点表示学到的参数值。从图中可以看出，第一个玩家将棋子放置在b6、f5、g5，或第二个玩家把棋子放置在c4、e4时，模型倾向于做出第1个玩家赢得游戏的预测；而第一个玩家将棋子放置在d3、e3、g4，或第二个玩家将棋子放置在a1、c6、e1时，模型倾向于做出第2个玩家赢得游戏的预测。在第一个玩家放置在b5、e3，第二个玩家放置在d6、e1时，更可能平局。

5.1.2 CART模型

对于CART模型，则利用graphviz库绘制最佳效果的决策树，由于最终高度为17的决策树表现效果最佳，绘制这么大的决策树存在困难，我们此处仅展示该决策树的前4层如下图所示。



从图中可以看出，在根节点处，按照a1处玩家1落子(a1=1)与(玩家2落子，空)(a1<1)对数据集进行了划分，而左右两支分别按照g1和d1处的情况进行了划分。我们发现，前4层基本都在判断棋盘的最下方3行，这与数据集中前3行棋子分布较多有关，同时也一定程度体现了下棋过程的逻辑。

深入观察各个节点，我们发现，{a1=1, d1=-1, b1=-1, d2<1, c2<1}情况下玩家2赢的概率较高，而{a1<1, g1<1, d3=-1, c3=-1, b1<1}，{a1<1, g1<1, c3=-1, c2<1, d3=-1}情况下玩家1赢的概率较高。同时，在前4层中，几乎没有任何一个节点处能对某一类进行完全的划分，这也体现了本分类任务的困难。

以上分析中我们发现，直接对模型参数进行可视化能够对模型产生初步感知，但未能得到较为明确的模型解释，以至于提取出全局的策略规则，尤其是决策树的模型，由于该树的深度较大，使得从决策树的路径中提取规则这一简单的想法会产生过多、过于繁杂的策略规则，甚至难以实施。因此，以下我们考虑借助其他工具继续发掘模型的解释性。

5.2 SHAP特征分析

在本节中，SHAP(SHapley Additive exPlanations)方法[7]是一种用于解释机器学习模型输出的通用方法，可以用来解释任何模型的输出。具体而言，在该方法中，将博弈论理论中的Shapley values应用于特征的重要性分析，把所有特征看做对最终模型输出的“贡献者”。在局部解释中，对于每个预测样本，SHAP值表示该样本中每个特征分配到的数值，即这个特征对于样本判断的贡献；在全局解释中，通过对所有预测样本计算每个特征SHAP值的各种统计量，理解模型的整体表现和结构。

5.2.1 局部解释

在局部解释中，我们随机从数据集中抽样两个实例进行SHAP值分析，观察线性回归模型和CART决策树模型分别是如何做出判断的。

5.2.1.1 实例1

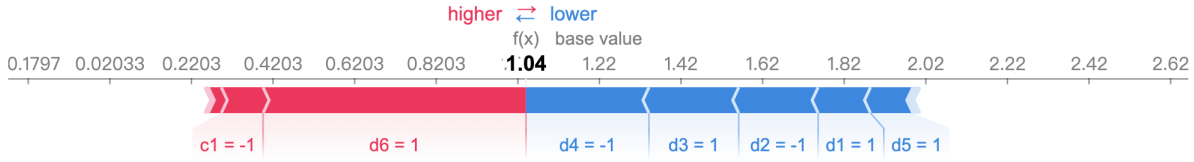
实例1中玩家1取得了胜利，两个模型均做出了正确的预测，实例1的落子位置、label和两个模型的预测结果如下所示

玩家1落子位置	玩家2落子位置	label	线性回归	CART

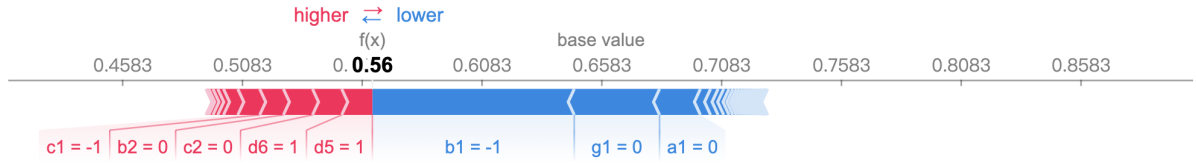
d1, d3, d5, d6	b1, c1, d2, d4	1	1	1

在两个模型中分别计算该实例的SHAP值，可视化表示如下所示。其中，base value为该类别的平均预测值，f(x)为模型对该类别最终的预测结果，蓝色和红色分别表示不同特征对最终预测的影响，越长表示该特征的影响越大。

线性回归：



CART决策树：



整体而言，两个模型在该实例处的预测都从类别平均值向混淆方向移动了，表示两个模型对该实例预测存在一定困难。线性回归模型中，d2、d4处玩家2落子，d1、d3、d5处玩家1落子，都导致了混淆，而d6处玩家1落子、c1处玩家2落子，则保证了最终预测的准确。CART决策树中，b1处玩家2落子、g1和a1处为空，导致混淆，但c1处玩家2落子，d5、d6处玩家1落子，以及b2和c2处为空则支持了最终的预测。

有趣的是，同样的特征d5=1在两个模型中有相反方向的影响，表示两个模型的解释存在不同。

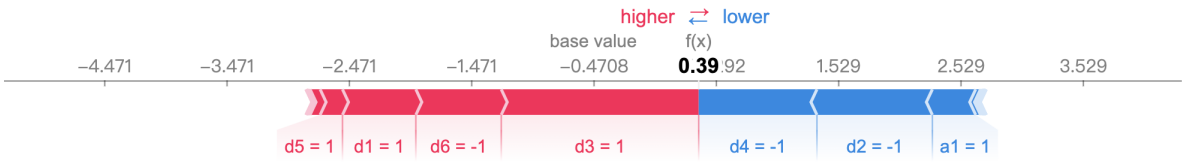
5.2.1.2 实例2

实例2中玩家2取得了胜利，线性回归模型做出了错误的预测（判断为1，即玩家1胜利）而决策树做出了正确的预测，实例2的落子位置、label和两个模型的预测结果如下所示

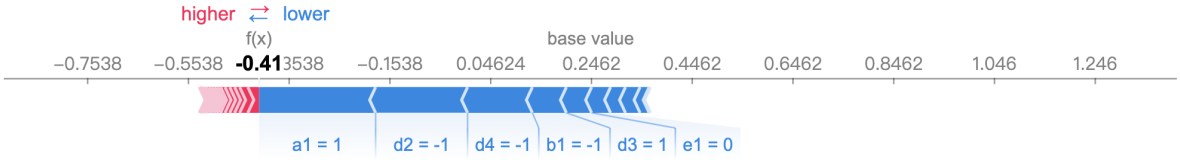
玩家1落子位置	玩家2落子位置	label	LR	CART
a1, d1, d3, d5	b1, d2, d4, d6	-1	1	-1

在两个模型中分别计算该实例的SHAP值，可视化表示如下所示，说明与5.2.1.1相同。

线性回归：



CART决策树：

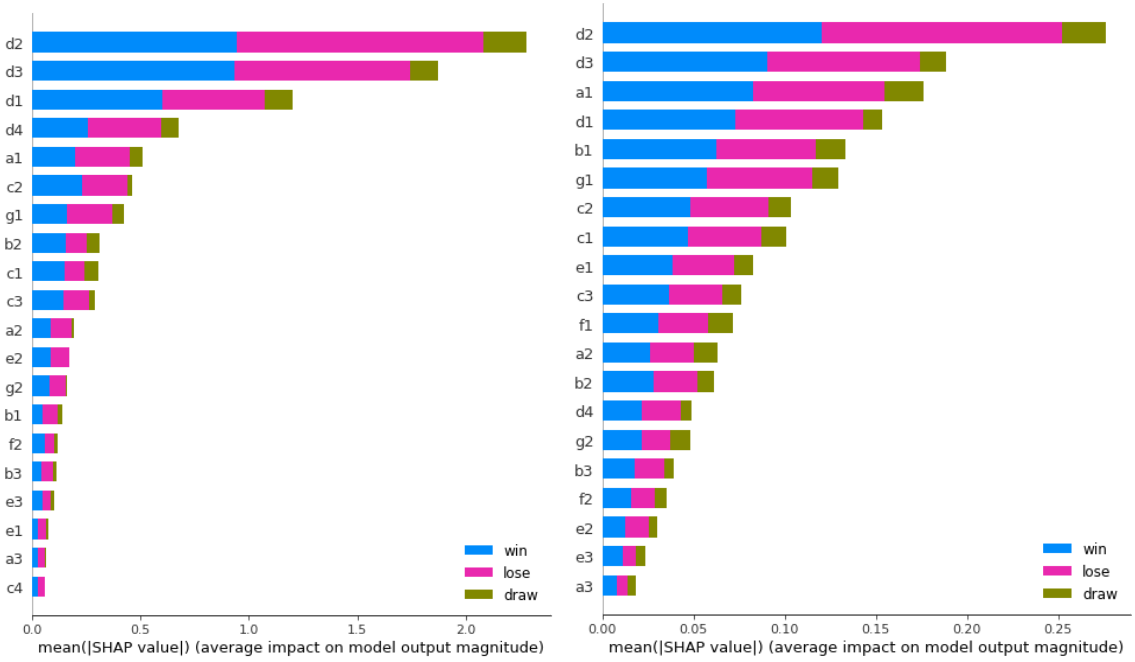


整体而言，线性回归模型在该实例处的预测从类别平均值向混淆方向移动，并导致最终预测错误，而CART决策树在在该实例处的预测却从类别平均值向支持方向移动，并给出了正确的预测。线性回归中，玩家1在d1、d3、d5处落子，玩家2在d6处落子导致了错误的判断。而CART决策树中，多个特征均支持正确的判断。

从两个实例中可以发现，线性回归模型只考虑落子(特征值为1或-1)的情况，而CART则还会考虑空位置对最终结果的影响。从模型原理出发，这个结果是自然的。这也是CART表现比线性回归更好的原因。

5.2.2 全局解释

首先分析模型整体的特征重要性，由于样本数量过大计算时间较长，随机抽取5000个样本进行分析。具体而言，计算5000个样本各个特征的SHAP值的绝对值均值，并由高至低排序，绘图如下所示。左图为线性回归模型的特征重要性，右图为决策树模型的特征重要性。

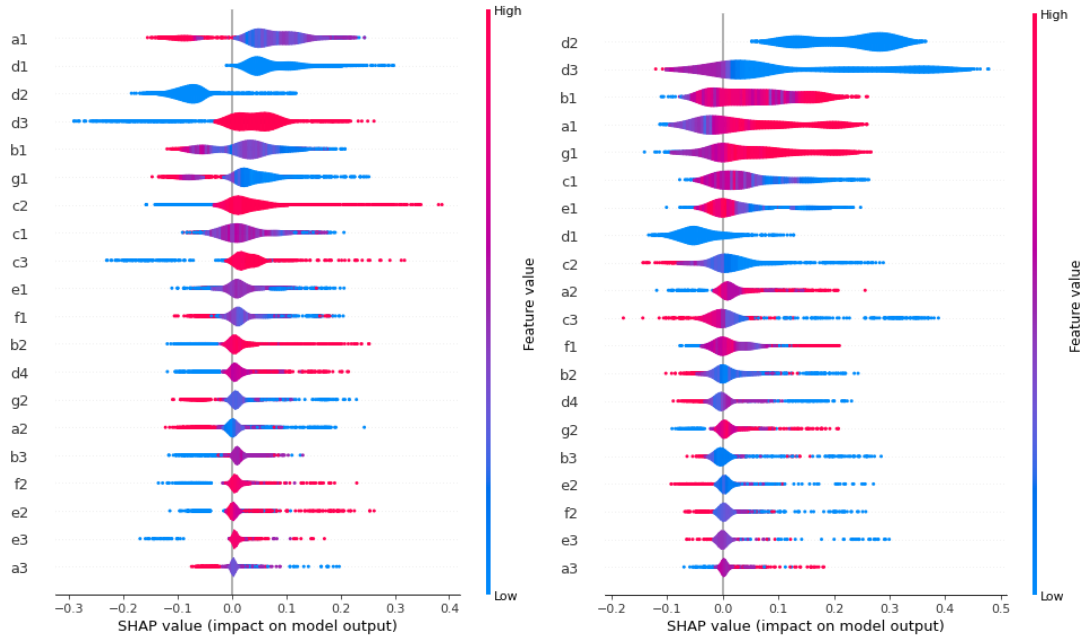


从图中可见，两个模型的重要特征top2都是d2和d3，且在win和lose两种情况下重要性均较高（draw的实例较少，暂时忽略）。但是从第三个特征开始，两个模型就存在较大差别。比较特征的SHAP绝对值大小，发现在线性回归模型中，前4个特征的shap都大于0.5，在决策中占据主导地位；而在CART决策树中，只有第一个特征shap大于0.25，各个特征重要性差别较小。由此可见，线性回归模型存在简化问题的趋势，部分解释了线性回归模型效果较差的原因。

由于CART决策树的模型效果比线性回归更好，以下使用CART决策树得到全局解释，并提取较为明显的策略规则。

依旧使用SHAP分析方法，绘制玩家1和玩家2赢得游戏的情况对应的SHAP值密度violin图。其中，特征的排序是按照SHAP绝对值的均值由高至低排序，特征越靠前，表示该特征对模型做出判断的影响越大。violin图每个bar对应位置表示SHAP值的大小，靠右的bar表示该特征的SHAP值的实际大小，SHAP为正值表示该特征支持做出当前判断，否则表示该特征不支持当前判断。bar的颜色则表示特征值的大小，红色为特征值较高（在本问题中为1，即玩家1落子），蓝色为特征值较低（在本问题中为-1，即玩家2落子）。

玩家1获胜和玩家2获胜的SHAP值密度分布图分别如下所示，左图为玩家1获胜，右图为玩家2获胜：



从图中可以分析出以下几条策略，

1）玩家1：

a. 在d3、c2、c3、b2处落子，会增加赢的概率；

b. 尽量不要在a1、g1、g2、a2、a3处落子；

2）玩家2：

a. 在d2、d3、c2处落子，会增加赢的概率；

b. 尽量不要在d1、g1、a2处落子。

由于全部满足某个策略的实例可能不存在，我们由以上策略，生成满足部分策略的子策略，在全量数据集中，符合策略的实例属于各类别的数量如下所示

策略	子策略	实例数量	玩家1获胜数量	玩家2获胜数量	平局数量
1)a	玩家1在d3、c2、c3落子	44	44	0	0
1)a	玩家1在d3、c2落子	624	568	48	8
违反1)b	玩家1在a1、g1、g2、a2落子	63	9	47	7
违反1)b	玩家1在a1、g1、g2落子	630	170	380	80
2)a	玩家2在d2、d3、c2落子	113	30	78	5
2)a	玩家2在d2、d3落子	1374	479	830	65
违反2)b	玩家2在d1、g1、a2落子	629	449	110	70
违反2)b	玩家2在d1、a2落子	3793	2336	1002	455

考虑到在原始数据集中，玩家1获胜的实例是玩家2获胜的实例数的约2.5倍，在我们筛选出的几条策略中，2)b效果不明显，但其他3条策略均有较好的表现。

5.3 模型可解释性量化比较

通过以上的分析，我们得到了两个可解释模型中不同特征的重要性。受到统计学中逐步回归法

（stepwise regression）[8]和Lundberg等人关于树模型可解释性工作[5]的启发，我们提出了如下量化比较可解释性的指标

1）重要特征贡献指标(Important Feature Contribution, IFC@k): 将特征按照模型在样本上的SHAP绝对值的均值降序排序（如5.2.2中展示）, 得到前k个特征称为k-重要特征。IFC@k即为使用k-重要特征预测的f1占全部特征预测的f1的比例，该值越高，表明该模型学习到的重要特征越准确。

2）特征重要性稳定性指标(Feature Importance Stability, FIS@k): k-重要特征的定义与IFC相同。FIS@k表示使用*k-重要特征*学习的新模型中，k-重要特征的排序列表与原始排序列表的秩的肯德尔秩相关系数（Kendall Rank correlation）：

$$\tau = P(C) - P(D) = \frac{C}{N} - \frac{D}{N} = \frac{C - D}{N}$$

其中， C是在两个列表中相对顺序保持一致的元素对数量；D是在两个列表中相对顺序不一致的元素对数量，N=k。该值越高，表示学习到的重要特征越稳定，即这些特征越能够代表数据集的性质。

实验设定与第4.1节相同，得到实验结果如下所示：

模型	IFC@3	IFC@5	IFC@10	FIS@5
线性回归	0.745(f1:0.367)	0.822(f1:0.407)	0.912(f1: 0.453)	0.2
CART决策树	0.627(f1:0.388)	0.773(f1:0.447)	0.903(f1: 0.533)	0.8

比较两个模型的结果，总体而言，线性回归模型在IFC指标上表现稍好，而CART决策树在FIS上表现更佳。

在IFC指标上，两个模型随着加入特征数量的增加，模型效果均有所提升。线性回归模型仅用3个最重要的指标就能达到全部特征f1值的75%，一方面，是因为线性回归模型本身f1值较低，另一方面在5.2节的特征重要性分析中也可以发现，线性回归模型前几个特征的重要性远高于其他特征，因此仅使用靠前的特征已经能达到较高的准确率。CART决策树使用前k个特征时，绝对f1值均高于线性回归，但IFC稍低，这与5.2节展示的结果，决策树特征重要性较均衡是一致的。

在FIS指标上，CART决策树远高于线性回归，表明CART学到的特征重要性较为稳定，而线性回归的特征重要性则随着数据集的变化产生较大变化，学习到的特征规则不具有普遍意义。

6. 总结

在本课题中，我们使用了两种可解释的分类模型（线性回归和决策树）以及三种不可解释的分类模型（随机森林、梯度提升树和多层感知机），依据四子棋棋盘在游戏进行中的状态，预测该棋局最终的胜负情况。

我们首先介绍了各模型的原理，并对比了模型的分类性能，发现梯度提升树的效果最好，且模型的稳定性也较好，多层感知机模型虽然分类效果较好，但是模型稳定性较差，而简单的线性回归模型由于无法拟合特征非线性关系，效果最差。

然后，在此基础上，我们针对两种可解释模型进行了可解释性分析，我们观测了模型所学习的特征权重，再利用SHAP方法对模型的输出进行解释，最终结合两者，基于逐步回归法提出了量化评价模型可解释性的指标IFC和FIS。我们发现线性回归模型学习到的重要特征相比于决策树更准确，但是决策树学到的重要特征更稳定，更具有普遍意义。

经过本次实验，我们首先复习了学习到的各分类模型的原理，然后通过在真实数据集上实践，了解了各模型之间的优劣，最后在可解释性分析的环节，加上了我们自己创新性的思考，收获颇丰。

参考文献

[1] https://www.boardgamegeek.com/filepage/40272/rules-simultaneous-connect-four

[2] https://en.wikipedia.org/wiki/Connect_Four

[3] http://archive.ics.uci.edu/ml/datasets/connect-4

[4] Hoffman R R, Mueller S T, Klein G, et al. Metrics for explainable AI: Challenges and prospects[J]. arXiv preprint arXiv:1812.04608, 2018.

[5] Lundberg S M, Erion G, Chen H, et al. From local explanations to global understanding with explainable AI for trees[J]. Nature machine intelligence, 2020, 2(1): 2522-5839.

[6] Doshi-Velez, Finale, and Been Kim. "Towards a rigorous science of interpretable machine learning."arXiv preprint arXiv:1702.08608(2017).

[7] Lundberg S M, Lee S I. A unified approach to interpreting model predictions[C]//Advances in neural information processing systems. 2017: 4765-4774.

[8] Efroymson,M. A. (1960) "Multiple regression analysis," Mathematical Methods for Digital Computers, Ralston A. and Wilf,H. S., (eds.), Wiley, New York.

分工

刘布楼：组织讨论及分工，数据集划分，不可解释模型开发，分类结果分析；报告相关部分撰写。

李佳玉：可解释模型开发，可解释性结果可视化分析，SHAP特征重要性分析；报告相关部分撰写。

苑苑：数据分析，可解释性量化指标调研，提出可解释性指标；报告相关部分及其他剩余部分撰写。