# 目录

[目录 1](#_Toc1608040667)

[说明 2](#_Toc165803597)

[第一部分：相关文献总结 3](#_Toc932299991)

[主要参考文献 3](#_Toc582316070)

[文献中的问题 3](#_Toc1844955206)

[第二部分：机器学习相关算法结果分析 5](#_Toc581039880)

[模型概述 5](#_Toc44271547)

[神经网络 5](#_Toc201192346)

[KNN 8](#_Toc1254079598)

[树模型 8](#_Toc525758309)

[实验设置 10](#_Toc383414503)

[数据集概述及预处理 11](#_Toc888201286)

[输出变量分布和相关性研究 11](#_Toc2077788410)

[缺失值统计与处理 14](#_Toc91814724)

[输入变量预处理 14](#_Toc1668287670)

[去量纲化 15](#_Toc2082819190)

[结果分析 15](#_Toc438043427)

[主要结果分析 15](#_Toc1877027116)

[对比实验 19](#_Toc1056045899)

[是否利用网格搜索调参 20](#_Toc636317899)

[是否进行one-hot encoding 20](#_Toc826845613)

[是否对数处理的影响 21](#_Toc100612783)

[均值填充与KNN填充的预测效果对比 22](#_Toc1212317949)

[是否添加batch normalization(BN)层 24](#_Toc1991042619)

[同时添加batch normalization(BN)层和drop-out 层 27](#_Toc687179856)

[预测效果随着网络层数增加的变化 27](#_Toc1300996705)

[预测效果随着网络神经元个数增加的变化 28](#_Toc1889753809)

[第三部分：变量选择的简单探究 29](#_Toc2006105243)

[第四部分：加入亲缘关系相关矩阵Z训练数据的尝试 33](#_Toc2016257715)

# 

# 说明

本文档主要分为两部分。第一部分是相关文献总结，主要分析三篇机器学习在genomic selection 中应用的文献，分析它们如何在gs数据中应用机器学习算法。第二部分是机器学习相关算法在两个具体的gs数据集中的应用情况；我利用了好几个常规的、比较常用的机器学习算法拟合gs数据集，并从数据的统计特性出发，在没有生物背景知识的情况下，分析比较各种不同情况下，不同机器学习模型的适用情况。

# 第一部分：相关文献总结

## 主要参考文献

* A review of deep learning applications for genomic selection
* Deep Learning for Predicting Complex Traits in Spring Wheat Breeding Program
* Multitrait machine- and deep-learning models for genomic selection using spectral information in a wheat breeding program

## 文献中的问题

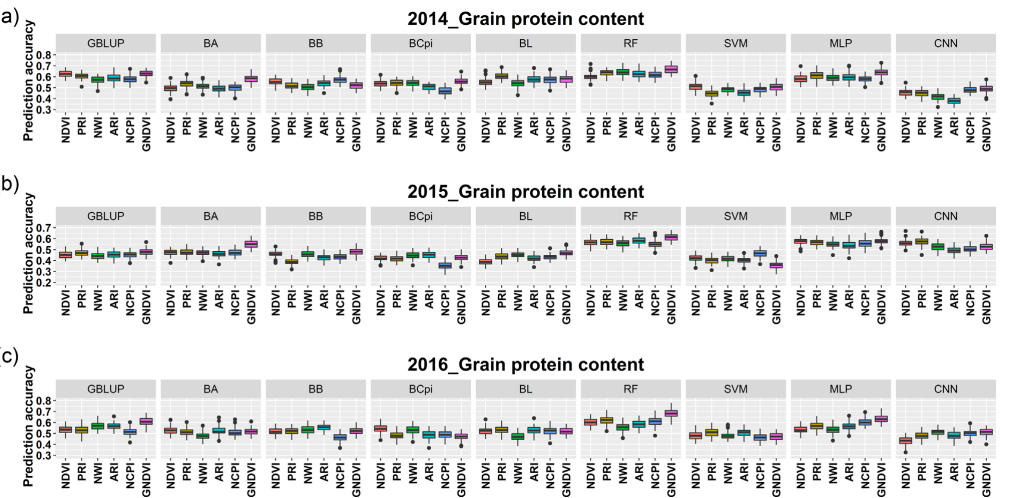
首先根据机器学习问题类型的分类，gs问题属于典型的回归问题：即用SNP makers 去预测各种我们关心的 traits，比如 grain yield, grain protein content, test weight等。“回归拟合”在机器学习领域是一个研究得比较深入的问题，有各种各样的模型可以用于回归拟合，比如线性回归、支持向量机、神经网络等。机器学习对于回归问题的研究主要是从线性模型（如线性回归、线性支持向量机等）过渡到非线性模型（典型的如神经网络）。

分析、总结不同的文献可以发现，在不同的情形下，机器学习算法和传统算法各有优劣，在什么情形下机器学习算法表现更好没有生物学情形下的详细分析。而在机器学习领域，样本量越大，越容易构建出优良的模型，模型过拟合的风险也越低。

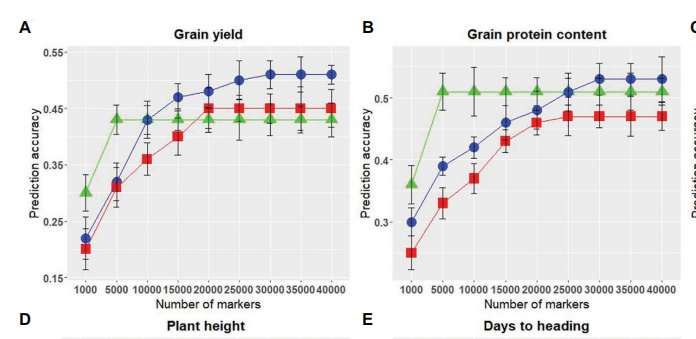
同时，可能由于文献作者是生物学背景出身而并非是数学或计算机背景出身，各种文献只是在大量地罗列试验结果，并着重比较了传统方法和机器学习方法的各种试验结果，并没有就结果作进一步地深入分析，并给出不同机器学习算法的适用背景。甚至，大部分作者也没有从生物学的角度对试验结果做很深入地分析。可能的原因是：机器学习方法很多都是black box方法，主要从数学的角度出发，并没有很好的可解释性；其次，文献作者可能对机器学习模型的原理了解不太深入，无法结合数学和生物学作进一步分析。

因此，我们最好能拿到具体的示例数据集，实际在数据集上运行各种机器学习算法，再分析比较各个算法的结果。

文献中大部分是在罗列试验结果：



文献中提到的其中一个情形需要特别留意： an increased marker number was related to improved prediction accuracy，a small number of QTLs which are controlling these traits。图如下所示：



也就是说在一定范围内，随着模型使用自变量的增多，模型的效果也会越来越好；并且少部分重要的自变量在很大程度上决定着模型的预测性能。在机器学习领域，有一个很重要的分支：变量选择，很可能对这个问题的研究大有裨益。变量选择算法的主要目的是寻找到对预测输出变量最为重要的输入变量，即对输入变量的重要性进行排序。最为经典的变量选择算法是LASSO模型，该模型主要在线性回归模型中通过添加一范数正则化使得权重变得稀疏：即不重要的变量权重全部变为0，重要变量的权重予以保留。学术界上常用的变量选择模型有：knockoff框架、梯度学习模型、稀疏可加模型、LassoNet模型等。这些模型主要是从数学的角度研究变量重要性，需要和生物学先验进行对比。

变量选择的主要参考文献如下：

* Controlling the false discovery rate via knockoffs
* Panning for Gold: Model-X Knockoffs for High-dimensional Controlled Variable Selection
* Learning Coordinate Covariances via Gradients
* sparse additive models
* LassoNet: A Neural Network with Feature Sparsity

# 第二部分：机器学习相关算法结果分析

## 模型概述

使用的机器学习算法如下：神经网络（NN）、卷积神经网络（CNN）、随机森林（RF）、K最近邻（KNN）、GradientBoosting（GB）、lightGBM、XGBoost、Cat\_boost。其中神经网络和卷积神经网络属于深度学习的范畴。其中KNN的取15个近邻，以树模型为基准的模型选800棵树。

### 神经网络

全连接神经网络（也叫多层感知机，MLP）拟合的是X与Y之间的非线性关系，而非线性关系主要是通过激活函数来体现。一般地，对于线性回归模型，有如下公式：

（1）

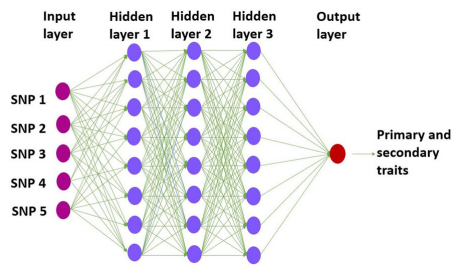
其中W是需要学习的参数，是误差项。

而有一个隐含层的神经网络可以用下述公示表示：

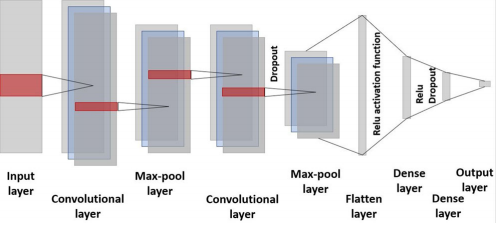
（2）

其中、是需要学习的参数，、是非线性函数。当、是线性函数时，式子（2）会发生退化，成为线性模型：

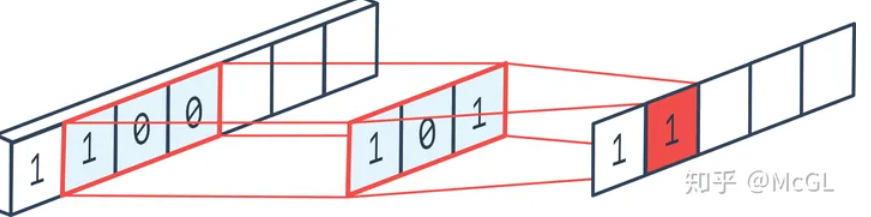
全连接网络的结构如下图所示，隐含层数和网络节点数可以自己设定，通过矩阵W的维度体现。下文有隐含层数和网络节点数对实验结果的具体分析。



卷积神经网络（CNN）是全连接神经网络的变体，只不过它的很多权重参数是共享的，这大大降低了需要学习的参数量。同时，通过卷积这个“滑动窗口”，有利于提取序列数据之间的空间信息，通过学习到相邻SNPs之间的关系辅助对Y的回归预测。相关结构如下图所示：



常用与序列模型的一维卷积操作：（实质上是滤波器的原理）



理论上说，神经网络的拟合、逼近能力是最强的，不管是线性还是非线性关系。因为根据数学上的The Universal Approximation Theorem，一个包含足够多隐层神经元的多层前馈网络，能以任意精度逼近任意的连续函数。但在实际应用中，神经网络的结果不一定就是最好的。因为神经网络有大量参数需要训练和学习，在样本量不足的情况下（根据我的经验，样本量最好大于500），很容易过拟合。其次，神经网络的结构需要自己去设计、调试，且超参数过多。一些微小的结构调整也可能带来结果上的巨大变化，这是一个漫长、冗余的过程。最后，神经网络不利于生物统计上常用的统计推断和假设检验，且预测过程是一个黑箱，很多时候无法得知各个输入变量的预测权重（最近变量选择的研究为破解这个问题带来了可能）。

但是，神经网络的优势也很明显。除了上文提到的逼近能力上的优势，还有如下优势：

* naturally capture, without the need to specify additional terms in the predictor (like interactions), nonadditive effects and complex relationships and interactions in large datasets.
* DL models more efficiently incorporate large numbers of omics data (Metabolomics, microbiomics, phenomics, Proteomics, Transcriptomics, etc.) in the same model, which is not possible with most machine learning and statistical learning method.
* CNN、LSTM等网络结构能自动提取序列不同节点之间的关联信息，以辅助预测。即very efficiently capture the correlation (special structure) between adjacent input variables, that is, linkage disequilibrium between nearby SNPs。

### KNN

KNN算法基于一个假设：相似的样本具有相似的特征。主要步骤如下：

1、计算待分类样本与训练集中每个样本之间的距离（通常使用欧氏距离或曼哈顿距离）。

2、选取距离最近的K个样本作为邻居。

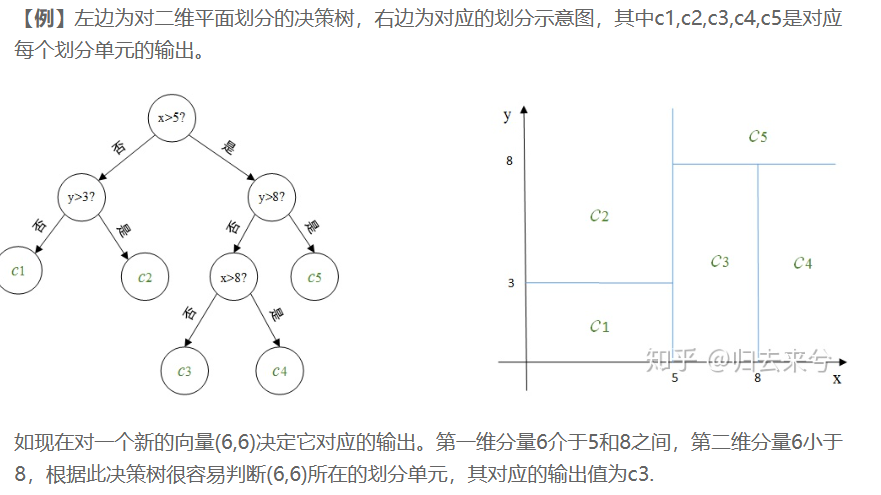
3、根据邻居样本的标签进行投票，计算邻居样本标签的平均值。

该算法原理相当简单，但是在样本量大的情况下时间复杂度较高。

### 树模型

随机森林（RF）、GradientBoosting（GB）、lightGBM、XGBoost、Cat\_boost都是以决策树模型为基础的集成模型。单棵决策树往往效果较差，通过对所有树的预测结果进行复杂加权，可以大大提升预测精度，这就是集成学习的意义。集成学习就是组合多个弱监督模型以期得到一个更好更全面的强监督模型，集成学习潜在的思想是即便某一个弱分类器得到了错误的预测，其他的弱分类器也可以将错误纠正回来。

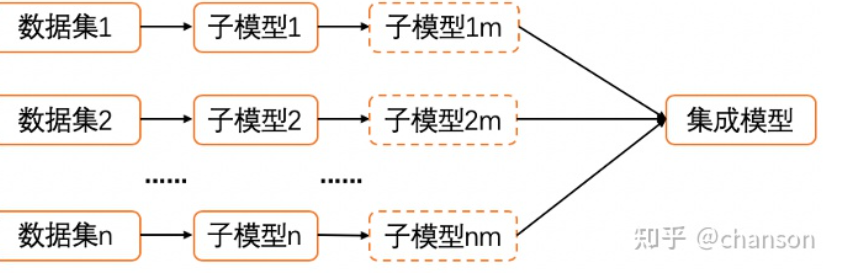
决策树回归的一个小案例如下：



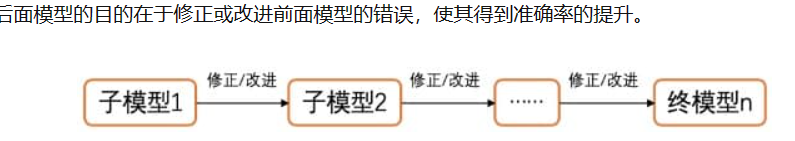
落在同一区域的样本都会被预测为同一个值。

集成模型主要有bagging和boosting方法，通过下面两个图很容易理解：

Bagging方法：如随机森林，主要是利用若干并行的模型进行预测，可以防止过拟合



boosting方法：主要用于防止欠拟合



最经典的Boost算法是AdaBoost。可以用“错题集”来类比。所谓“错题集”，就是搜集做错的题目，然后抽空把这些错题重新做一遍，看看是否真正掌握了这些错题。现在，让我们把这个错题集的做法扩展下。假设重做错题的时候，有些题又做错了，那我们可以再准备一本“错题集二”，专门收集这些重复做错的错题。以此类推，我们可以有“错题集三”、“错题集四”、“错题集五”……直到所有（或者，基本上所有）题目都在重做时做对了。这就是AdaBoost背后的直觉。首先运行一个模型，进行预测。然后再运行第二个模型（“错题集二”），拟合之前错误预测的实例（当然，这一过程伴随着相应权重的更新）。重复这一过程，直到模型能够拟合所有（或者，基本上所有）训练实例。

Gradient Boosting类似AdaBoost。AdaBoost拟合上一个模型预测错误的实例，梯度提升拟合上一个模型的残差（Residual Errors）。回顾一下之前错题集的类比。这次不收集错题了，而是查看下做错的题目是从哪一步开始做错的，然后从做错的这一步开始，重现编制一道新题。新题的解题步骤从之前做错的那一步开始。这样我们就得到了一种新形式的“错题集”。这种错题集收集的不是完整的错题，而是根据之前的错题重新编制的新题。然后，我们做一下这些新题，并再次查看其中做错的题目是从哪一步开始错的，然后据此重新编制新题。以此类推，直到所有（或者，基本上所有）题目都做对了。因此，在梯度提升中，首先运行一个模型，进行预测。然后，计算残差，让这些残差成为新的训练样本。接着用一个新模型去拟合这些残差，以此类推。最后，将这一过程中的所有模型加起来，就得到了最终模型。

另外，lightGBM、XGBoost、Cat\_boost都是gradient boosting的变种（或者说工程上的优化实现），主要在提升运行效率、提高精度、减少内存消耗等方面进行了改进。

## 实验设置

做10次以上的重复试验（大部分是12次，少部分是10次），每次都随机划分0.8的数据集进行训练，0.2用于测试，保存所有试验结果进行比较。

如果需要，则利用4折交叉验证选择超参数。但是大部分模型都是我根据经验在默认超参数附近范围选择超参数，这种情况下效果反而更好。因为交叉验证也是需要指定范围，这反而在一定程度上限制了参数选择的空间，导致效果没有达到最佳。

一般情况下，在机器学习领域，我们假定随机划分的训练集和测试集满足同一个分布；否则，模型的效果会出现偏差。

## 数据集概述及预处理

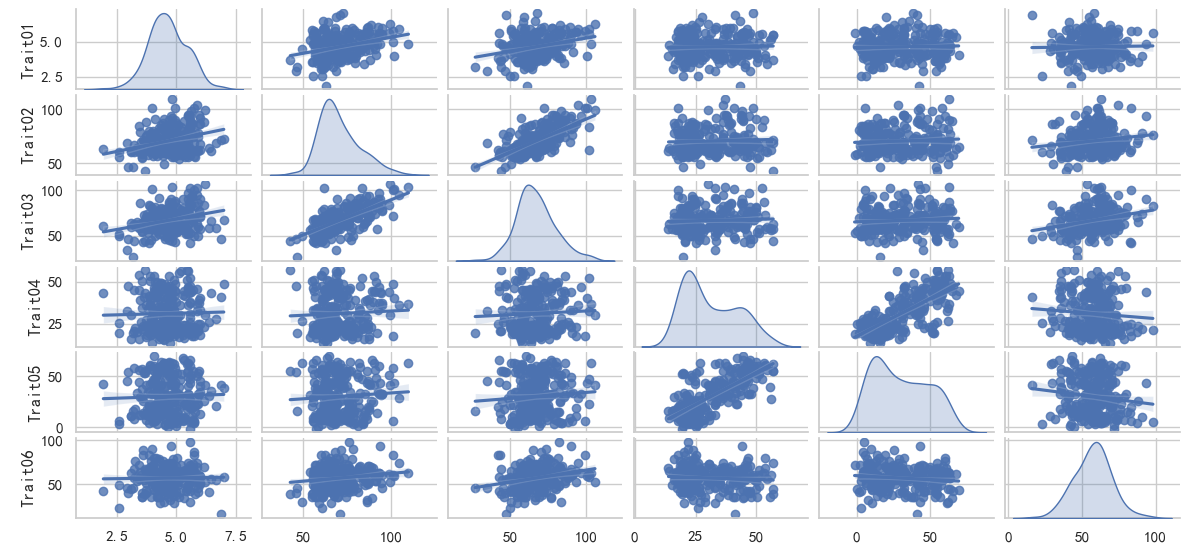
本文档主要使用两个数据集：

* 数据集1：Wheat dataset，共有250个样本2799个输入变量，需要预测trait01，trait02，trait03，trait04，trait05，trait06 六个输出变量。
* 数据集2：Forest dataset，共有1008个样本16382个输入变量，需要预测diameter、branchcluster、straightness三个输出变量。

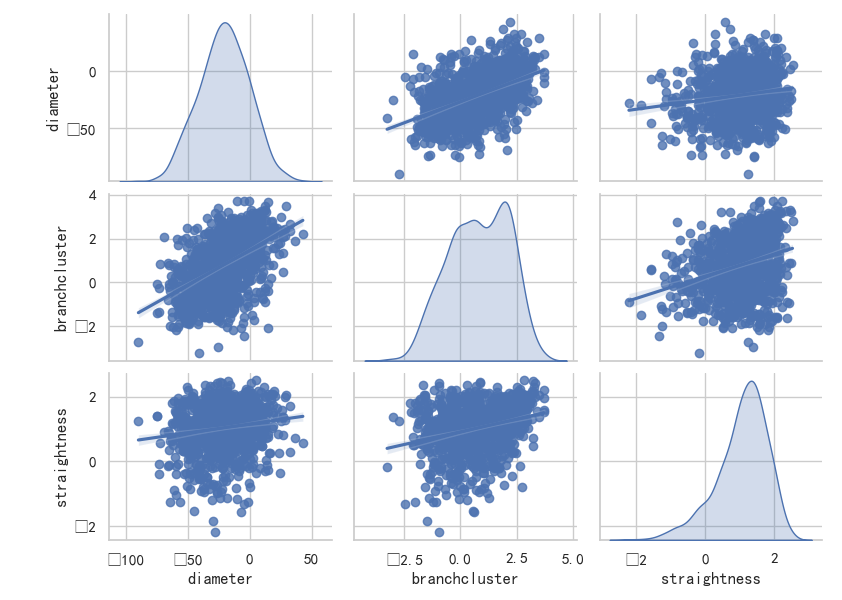
由于时间关系，大部分模型研究的是针对trait02和branchcluster的单变量输出回归拟合，很多不同处理和模型的对比也主要是在这两个输出上进行。

### 输出变量分布和相关性研究

数据集1：



数据集2：

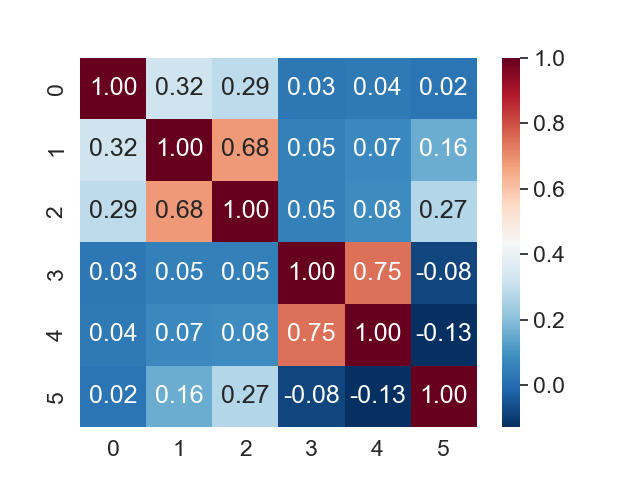


由图可以看出，数据集2的变量偏态情况不算严重，数据集1的变量trait04，trait05的偏态有一点严重，最好能做如下取对数处理：

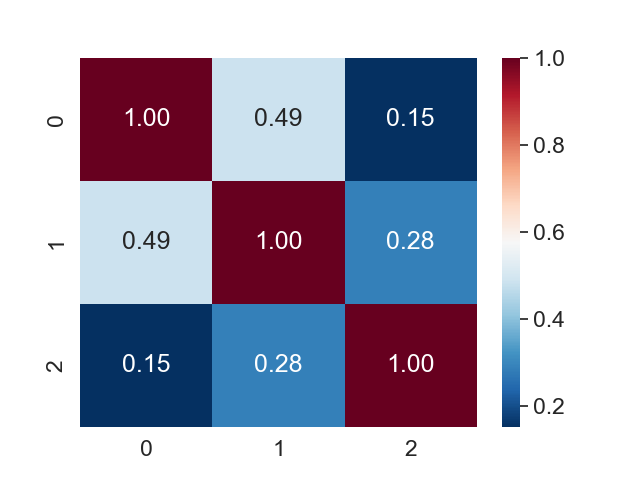
后面的结果展示中会对比trait04，trait05是否做取对数处理结果的差别。

数据集1、2的相关系数矩阵分别如下所示：

数据集1：



数据集2：



可以分析得出，数据集2不同输出变量的线性相关关系较弱，数据集1 第2、3个输出变量和第4、5个输出变量之间线性相关关系相对较强。线性相关关系的较强时，利用多输出预测模型可能对预测结果有所提升。下文会有详细实验结果进行对比分析。

但是，使用皮尔森相关系数依然有很大局限性，使用互信息能捕获变量之间的非线性相关关系可能会更加合理。

### 缺失值统计与处理

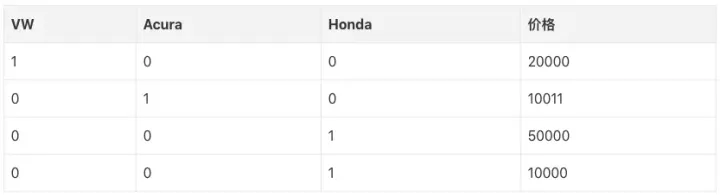
数据集2的输入变量中存在大量缺失值，当某输入变量缺失值的比例大于**0.1**时，将整列直接删除，否则利用算法进行填充。输入变量个数由16382减少至13514。这里主要采用KNN进行缺失值填充，下文也对比了均值填充和KNN填充时，模型预测结果的差异。数据集1的输出trait02中存在极少量的缺失，这里也直接利用相邻值进行填充。但是这在一定程度上是不合理的，下文也会对比填充和直接删除对预测结果的影响。

### 输入变量预处理

由于输入变量SNP输入标记变量（类别变量或离散变量），其值0，1，2之间没有数量上的大小关系，需要进行特殊处理。因此，我选用one-hot encoding 对输入变量进行预处理，处理完后数据维度被大大扩充了，这可能会极大提升时间花销，且输入矩阵会变得非常稀疏，存储花销也大大增大。

one-hot encoding小例子：





下文会对比输入变量是否做one-hot encoding对预测结果的影响分析。

由于输入变量的维度过高，考虑过利用PCA和T-sne进行降维。但是，由于SNP矩阵过于稀疏，降维后信息损失过大，会极大影响预测效果。故下文不再考虑降维。

### 去量纲化

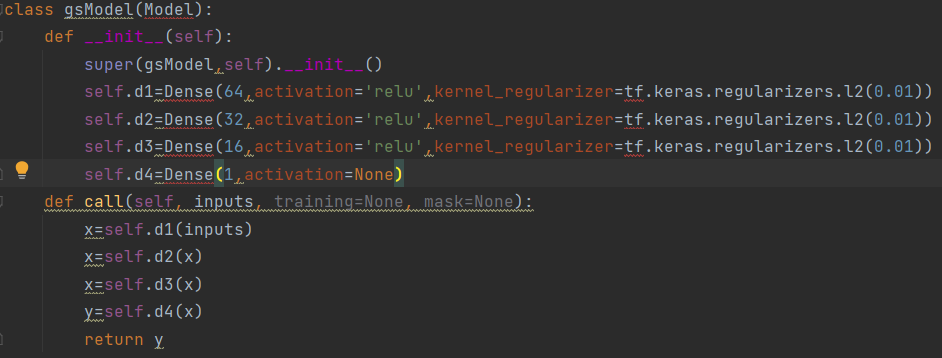
输入输出变量都进行如下归一化操作：

## 结果分析

本章主要分成两部分。第一部分是各种机器学习模型对于数据集1 trait02的预测分析和对比（主要是单输出模型），第二部分是对比各种预处理方法和不同的模型结构对结果的影响分析。

### 主要结果分析

NN结构：



该结构就是简单的全连接结构，又称作前向神经网络，需要学习大量参数，容易过拟合，但运行速度较快。

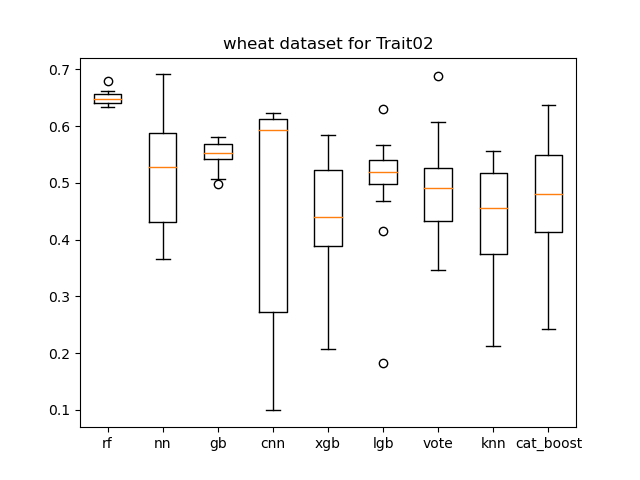
CNN结构：

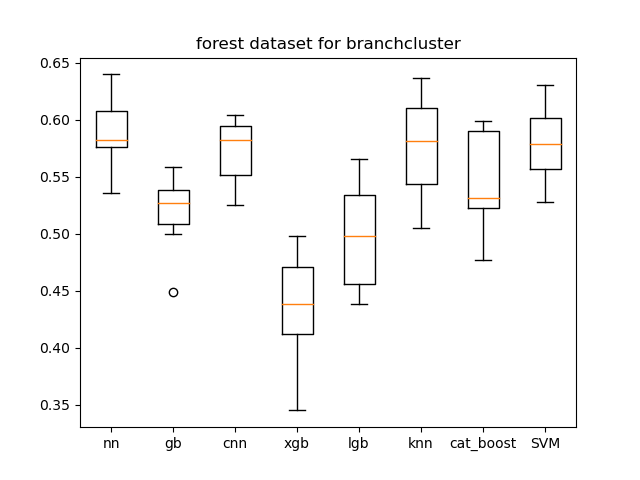


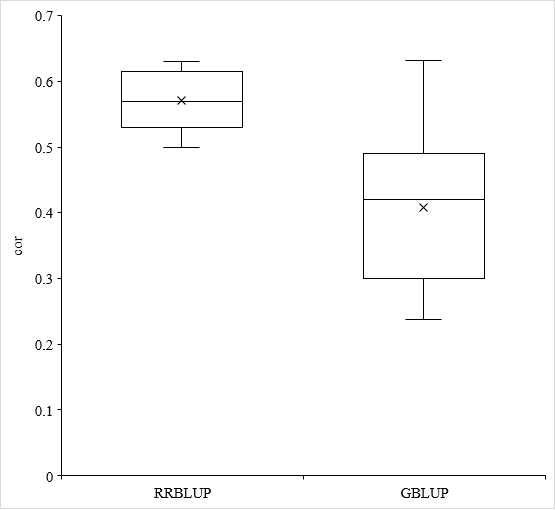
这是具有卷积结构的神经网络，参数大大减少，且能学习到序列之间的空间信息。需要注意的是，CNN需要输出变量是序列型的数据：（sample\_size, step\_size, dimension\_size）。为此，需要将数据转化为这样的结构：（sample\_size, SNP\_NUM, 1）。卷积是作用在第二个维度step\_size上的，CNN能充分利用SNP的空间结构信息进行辅助预测，在下文数据集2的实验结果充分证明了这一点。但是CNN的运行时间较长，且设计流程较为复杂。没有恰当的网络结构设计，很难得出好结果。

网络设计一般要根据数据量决定：样本数多，可以设计得稍微复杂；样本数少，应尽可能设计得简单。

测试集结果如下：（12次独立重复实验）







从上面的图可以看出，随机森林在数据集1中表现较好，全连接神经网络(NN)和gradient boosting (gb) 表现也比较稳定，但NN的波动相对较大。而CNN、lgb的波动性更大，这有可能是对输出变量的填充导致的学习差异。当输出变量有缺失值时，直接删除可能是更好的选择；因为在输出变量中存在缺失值可能使得模型往错误的方向上学习，误导模型。同时，250的样本量并不足以让神经网络得到很好地训练。

而在数据集2中，整体拟合效果优于数据集1。除了xgb效果较差外，其他模型的效果都相对较好。由于该数据集中输入变量个数太多，随机森林需要并行学习800棵数（类似于并联），时间花销巨大，只跑了两个结果：0.4619869626774553,0.5236560890637157。同时，NN和CNN的拟合结果较为优良，说明在大样本量的情况下，大量可学习参数使其强大的拟合能力得到体现。需要注意的是，在数据集2中，CNN的效果相比数据集1得到很大提升，波动也大大降低，说明CNN在拟合像SNP这种具有序列效应的模型有很大优势。而KNN这种比较简单的模型也取得了不错的效果。KNN是利用15个最相邻的样本的输出值进行加权平均；从先验的角度，当两个样本的SNP较为相近时，其表型（形状）也较为相近。这可能揭示出KNN效果较为良好的原因。为此，下文会具体探究KNN中随着neighbour个数的增加，模型预测效果的变化趋势。XGB中则存在大量防止过拟合的机制（防止训练集效果很好而测试集效果很差的机制），这可能导致其学习不充分；作为gb的变种，XGB效果反而更差。

利用rrBLUP方法的结果如下：

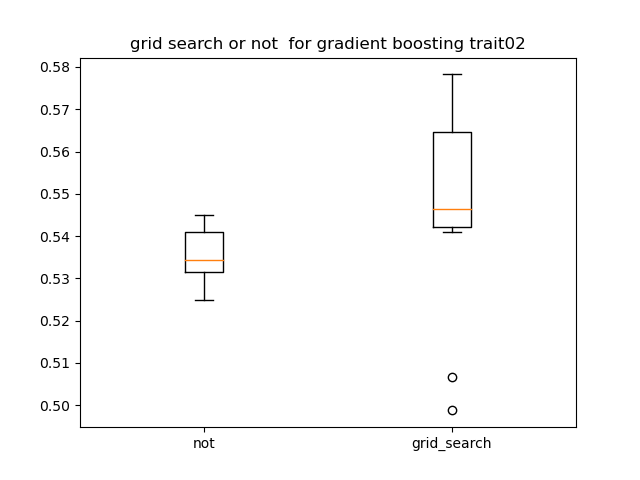
简单结论：在样本量充足时，首选CNN，可以利用SNPs之间的效应和超强的拟合能力提升预测效果；如果需要节约时间、模型简单，可以选CNN进行拟合，不仅效果较为准确，且符合遗传学先验；在样本量欠缺时，可以选择GB、SVM、KNN这三个模型，能保证精度同时时间花销不太大。

### 对比实验

主要进行了以下对比：

* 是否进行one-hot encoding的预测效果对比
* 均值填充与KNN填充的预测效果对比
* 利用网格搜索调参与直接默认参数附近调参的预测效果对比
* 数据集1中trait04，trait05是否取对数处理对预测结果的影响
* 单输出模型与多输出模型的预测效果对比
* KNN超参数对比：随着neighbour个数的增加，模型预测效果的变化趋势
* 全连接神经网络结构对比：是否添加batch normalization(BN)层
* 全连接神经网络结构对比：是否添加drop-out 层
* 全连接神经网络结构对比：是否同时添加batch normalization(BN)层和drop-out 层
* 全连接神经网络结构对比：预测效果随着网络层数增加的变化
* 全连接神经网络结构对比：预测效果随着网络神经元个数增加的变化
* KNN超参数对比：neighbour个数对模型预测效果的影响

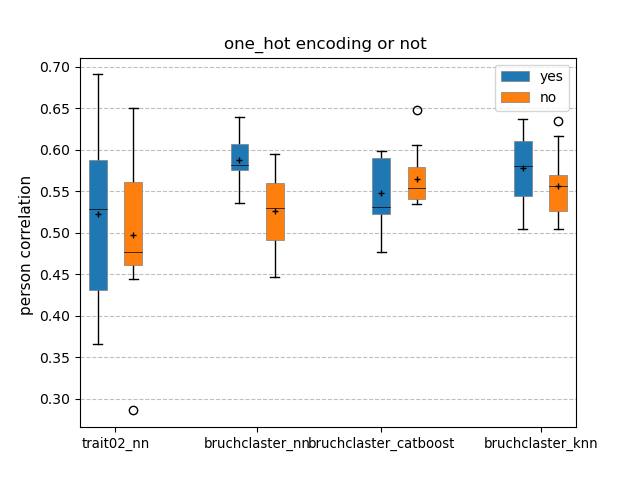
#### 是否利用网格搜索调参



从数值上看，差别不是那么大；利用网格搜索调参后能得到的最优值更大，但是却没有那么稳定。且超参数范围搜索范围同样很难界定，如果设定太大会大大拖累运行时间。同时由于对比实验较少，很难直观看出那种方法最好。但是根据我个人的经验，网格搜索调参对于数据集2耗时过长，不太适宜。

#### 是否进行one-hot encoding

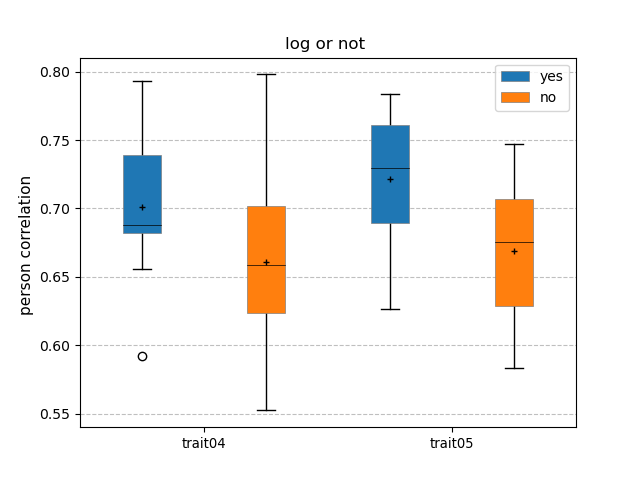
单输出模型、one-hot encoding、KNN填充 下的对比结果如下：



从总体上看，对离散变量进行one-hot encoding虽然增大了时间花销，但是对模型效果的提升依然较为显著（尤其是全连接神经网络），机器学习模型中还是比较推荐该做法。

#### 是否对数处理的影响

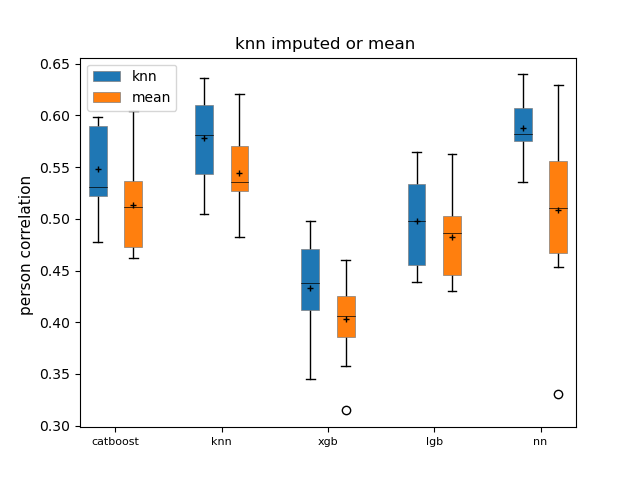
在多输出模型、one-hot encoding、KNN填充 下的对比结果如下：



取完对数后预测效果有显著提升。这证明对于偏态数据，需要进行取对数操作。

#### 均值填充与KNN填充的预测效果对比

单输出模型、one-hot encoding

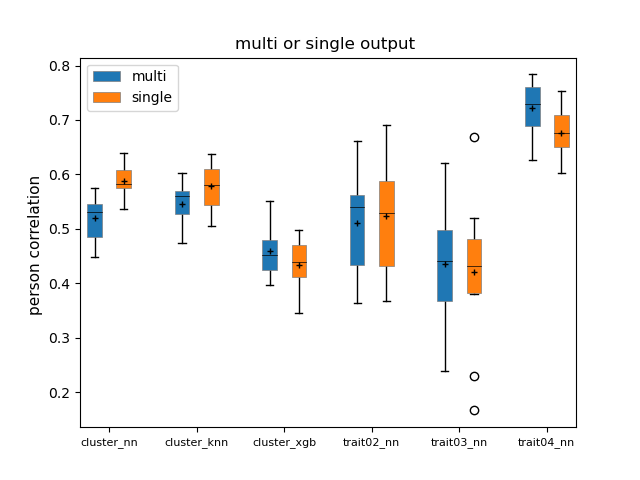


利用KNN填充后的预测效果明显好于只用均值填充，因此存在数据缺失时，应该优先考虑利用算法进行填充。常用的机器学习填充算法还有：随机森林填充（miss forest）、生成对抗网络填充（GAN）等。

**单输出模型与多输出模型的预测效果对比**

之所以要对比单输出模型和多输出模型的差异，是因为根据机器学习理论，多输出模型可能可以更好地利用输出变量之间的交互效应（线性或者非线性关系）辅助进行建模预测，以提升模型整体的预测效果（特别是对于神经网络而言）。但是，能否提升效果还是得根据数据的特性决定。

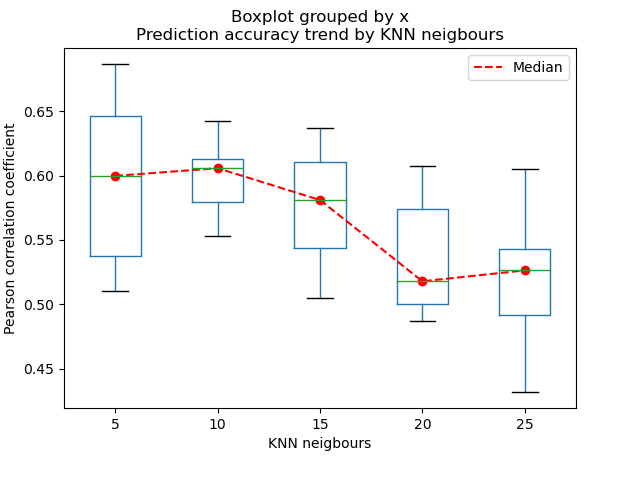
生物不同的表型（形状）之间应该是具有联系的，因此可以尝试探究多输出模型的特性，探究多输出模型相比单输出模型在预测精度上能否有提升。



不出意外，多输出对KNN的预测没有帮助，因为KNN只是在通过不同样本之间的距离寻找近邻样本，利用近邻样本取值进行预测，没有用到输出变量的交互效应；由于森林数据集中输出变量之间的相关关系较低，对预测的提升也没有帮助。由于trait03、trait04之间的相关系数高达0.74，因此多输出能在一定程度上提升预测效果。

但是，当神经网络由单输出改成多输出后，调参的难道会大大上升，网络结构也更难设计。

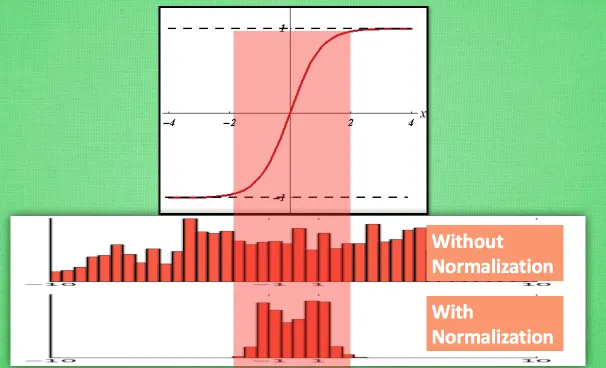
**KNN超参数对比：neighbour个数对模型预测效果的影响**

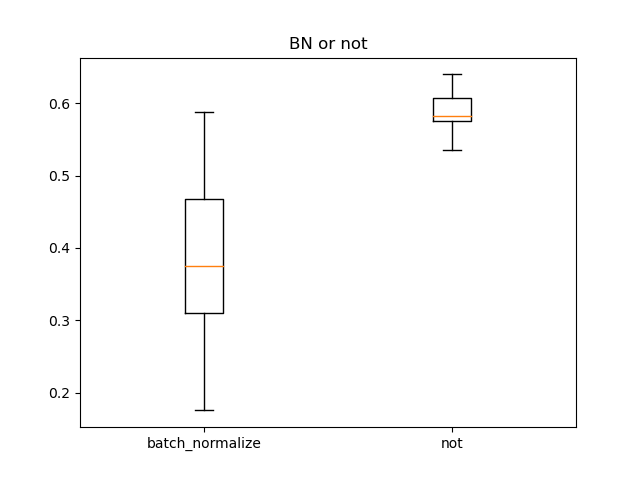
****

由图可以看出，近邻个数在10或者15时效果是最好的；取过多或者过少的近邻都会对模型产生不利，影响模型的预测准确性。

#### 是否添加batch normalization(BN)层

Batch normalization 的 batch 是批数据, 把数据分成小批小批进行 stochastic gradient descent. 而且在每批数据进行前向传递 forward propagation 的时候, 对每一层都进行 normalization 的处理。我们先有数据 X, 再添加全连接层, 全连接层的计算结果会经过 激励函数 成为下一层的输入, 接着重复之前的操作. Batch Normalization (BN) 就被添加在每一个全连接和激励函数之间。

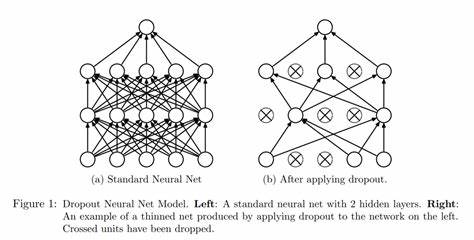


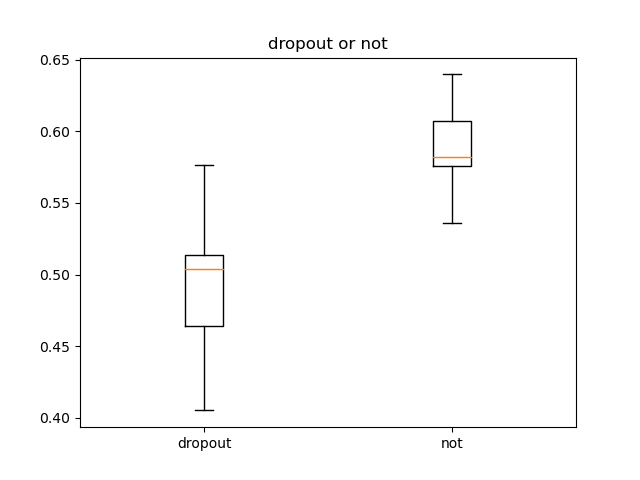


由于使用的是relu激活函数，不存在梯度消失的问题，batch\_normalization层无法提升预测性能。由于gs数据的最终目的是使得预测变量与真实变量的趋势一致，因此使得预测分布更为分散反而更能符合要求，而BN层却恰恰相反。

**是否添加drop-out 层**

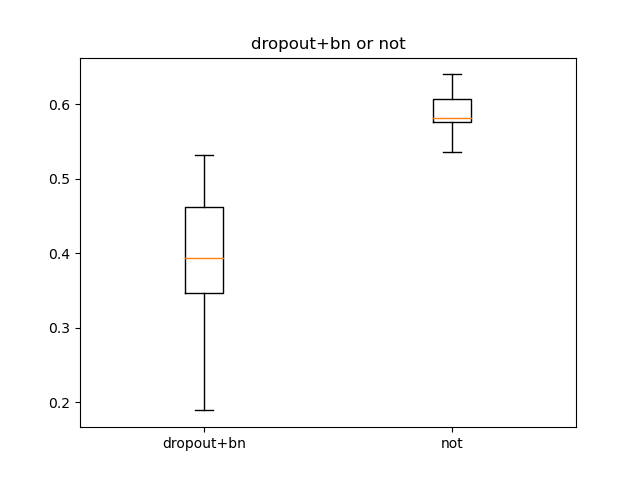
dropout是神经网络中的结构，用于防止过拟合。Dropout说的简单一点就是：在前向传播的时候，让某个神经元的激活值以一定的概率p停止工作，这样可以使模型泛化性更强，因为它不会太依赖某些局部的特征，如图所示。





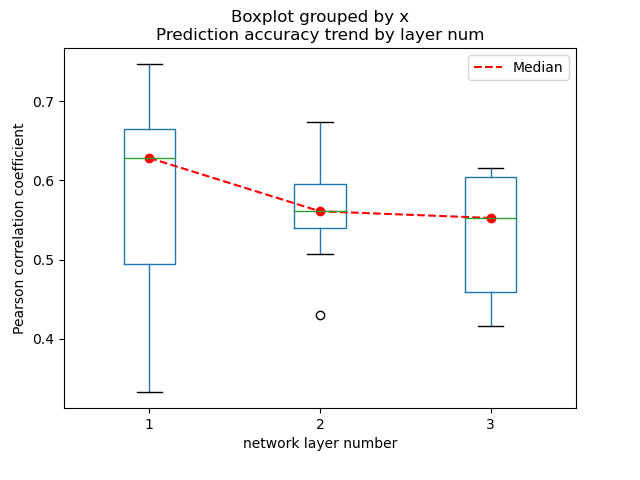
效果也不太明显。

#### 同时添加batch normalization(BN)层和drop-out 层



#### 预测效果随着网络层数增加的变化

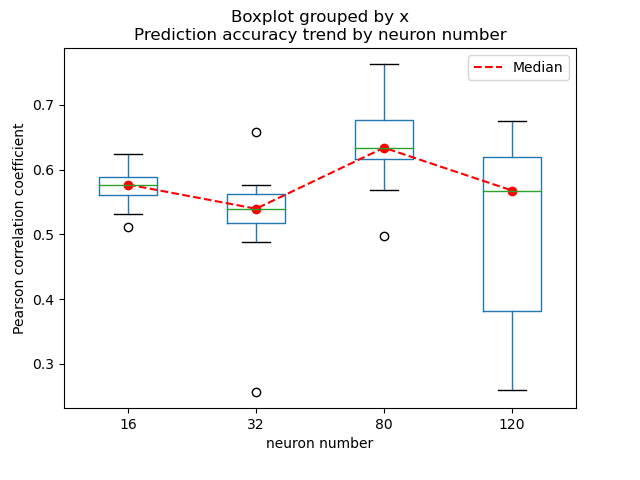
在one-hot encoding 、KNN填充、单输出模型下、没有添加dropout和batchNormalization层的情况下，10次独立重复试验的结果：



可以看出，随着网络层数的增加，预测精度变化不大；但是在网络层数=2时，神经网络预测的波动比较小，预测结果相对比较稳定；当网络层数为1时，模型的预测精度极其不稳定。因此，可以考虑将网络隐含层数设为2。

#### 预测效果随着网络神经元个数增加的变化

在one-hot encoding 、KNN填充、单输出模型下、单隐含层、没有添加dropout和batchNormalization层的情况下，10次独立重复试验的结果：



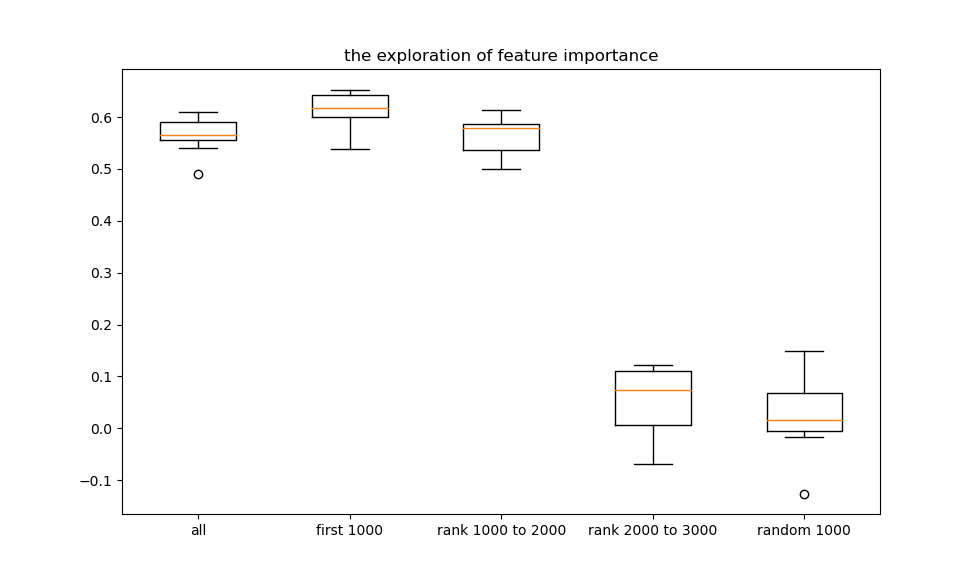
从图中可以看出，在隐含层神经元数为80时，取得了最好的结果。

从上面几个对神经网络结构探索的试验中可以发现，神经网络结构的设计是相对复杂的课题，没有什么显著的规律可言，需要从数据本身的特点出发。在样本数较少（经验上少于500）的情况下，使用神经网络不是一个好的选择，大量可学习参数会使得网络的预测结果产生剧烈的波动。神经网络结构的设计可遵循相对简洁的原则：隐含层数可以设为2，第一个隐含层神经元数可设为60至80，第二个隐含层神经元数可设为16至32，不添加dropout和BatchNormalization层即可。

# 第三部分：变量选择的简单探究

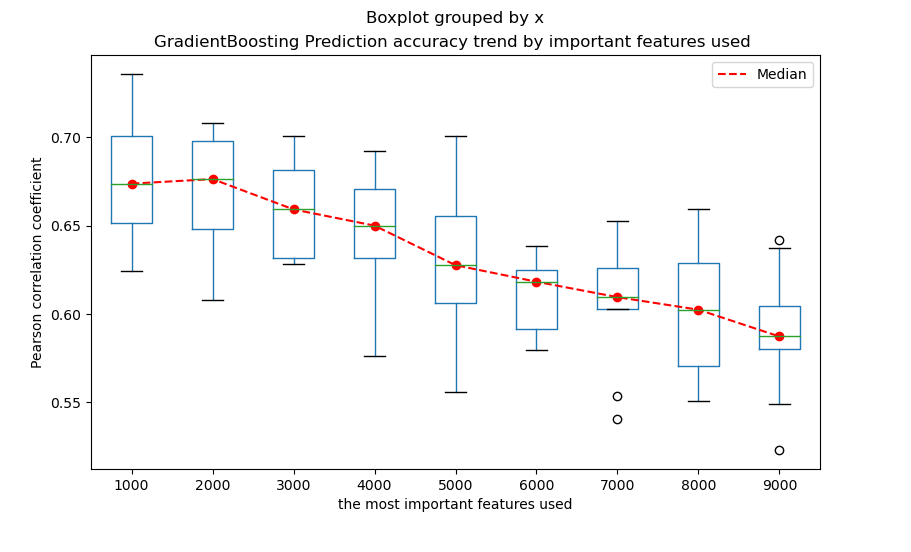
本部分利用树集成模型中的feature\_importance\_特性（具体计算方法比较复杂），对forest数据集中的变量重要性进行简单探究。

主要方法：首先，利用cat\_boost的feature\_importance\_接口给出所有输入变量的重要性描述；随后，分别取出变量重要性排在前1000的输入变量、排名1000至2000的输入变量、2000至3000的输入变量以及随机提取的1000个输入变量；将取出的变量分别进行模型拟合，并在测试集上进行测试，对预测分数进行探究。 利用KNN模型进行了10次独立重复试验，结果如下：



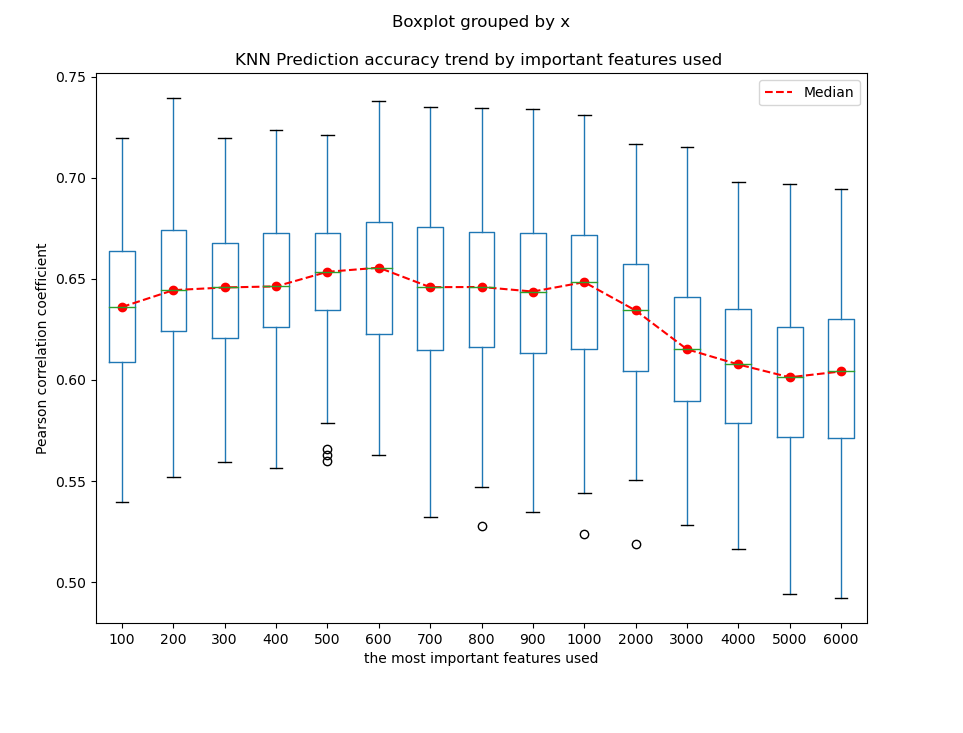
可以看出，选择的最重要的1000个变量拟合的模型得到了最佳的结果，其甚至比选择全部变量的效果还要更好，说明部分变量对模型具有干扰作用，会影响模型精度，这充分说明了变量选择的重要性；重要性排名1000至2000的变量结果变稍差，重要性排名2000至3000的变量预测精度开始大幅下降，与预期结果一致。即利用最重要的2000个输入变量就可以保证模型预测精度了。

利用GradientBoosting 继续探究模型预测精度与使用前n个最重要的变量的关系图如下：



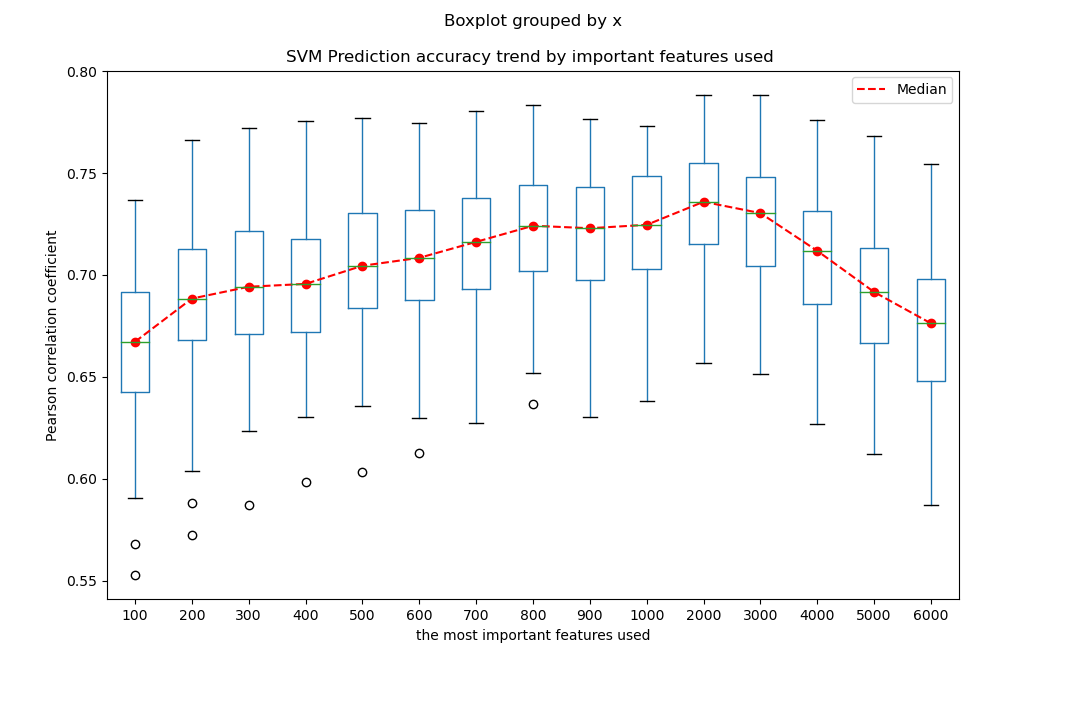
基本上使用前2000个重要变量就能使得预测比较精准，与上图结论一致。需要注意的是，由于试验时间较短，导致结果出现波动，与文献中的结果不太一致；且进行变量选择的模型是否精准对最终结果的影响也很大。同时，考虑模型构建、训练时的误差也不容小觑。

利用KNN进行100次试验结果如下：



KNN模型比较特殊，是利用近邻样本进行加权预测；添加太多无关变量可能会对模型精度产生很大影响。

SVM独立重复试验100次的结果：



# 第四部分：加入亲缘关系相关矩阵Z训练数据的尝试

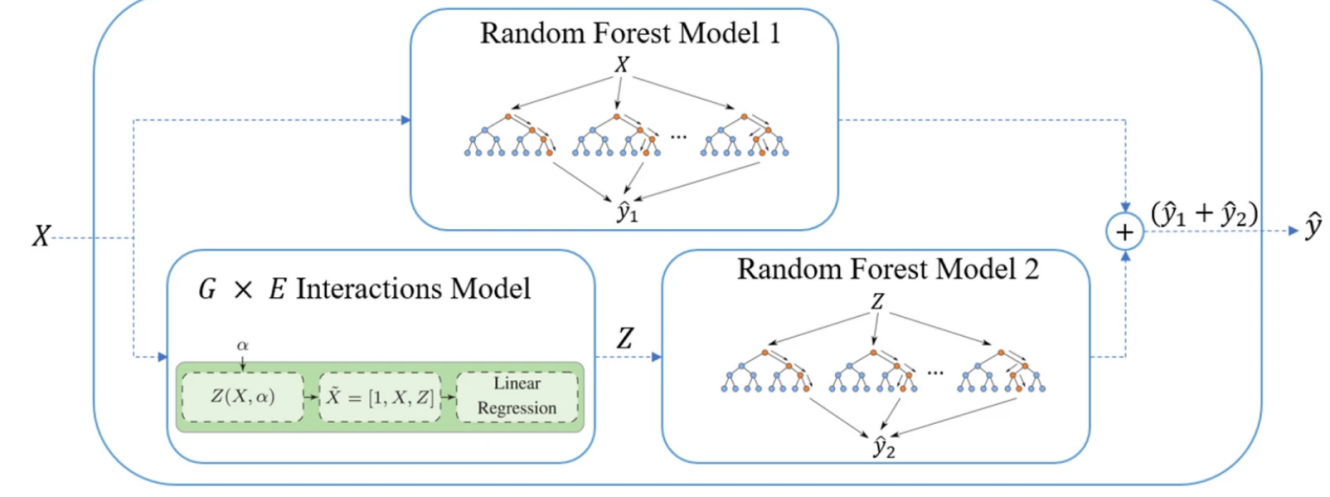
由前面的结果可以看出，除了神经网络（CNN、全连接）、KNN、SVM之外，其他机器学习模型的效果并没有比传统模型好多少，因此我想尝试分析一下原因，以提升模型预测性能。根据生物学传统上使用的混合线性模型：

y=Xb+Zu+e

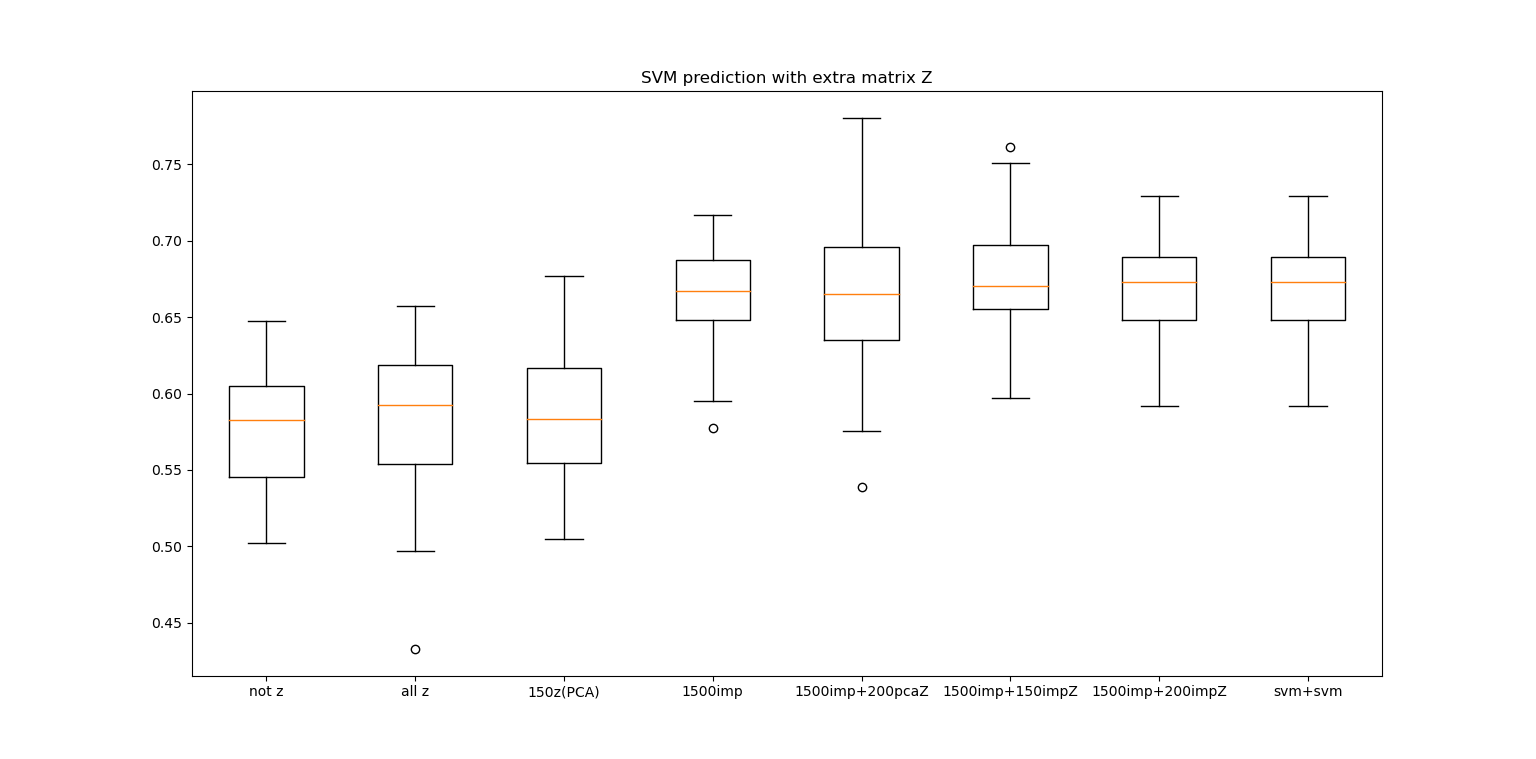
该模型除了使用数据（X,Y）之外，还额外使用了矩阵Z；该矩阵和亲缘关系矩阵是相关的。而之前尝试拟合机器学习模型时，只使用了数据（X,Y）。因此，可以尝试加入Z进行模型拟合。

可以尝试从以下角度进行对比试验：

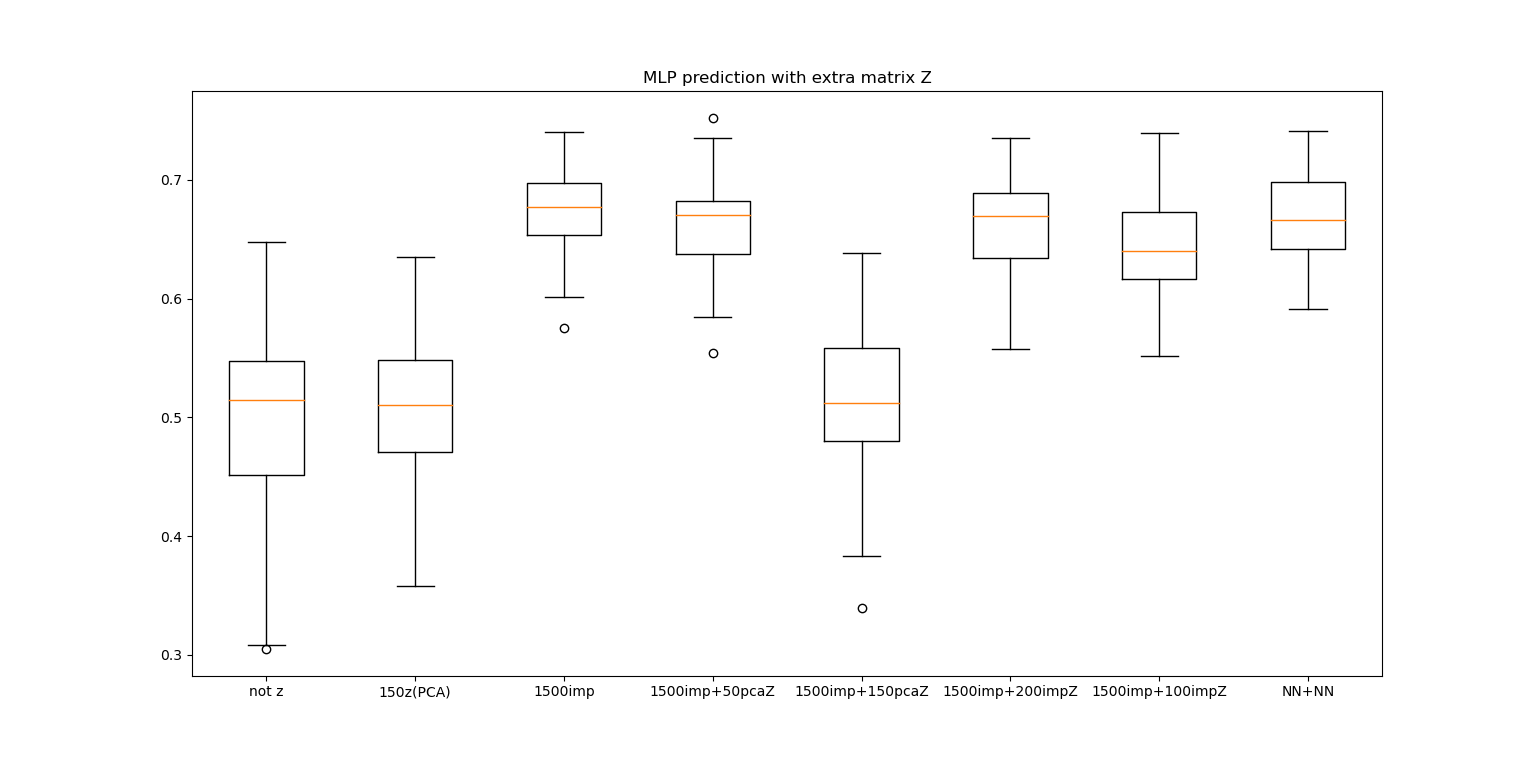
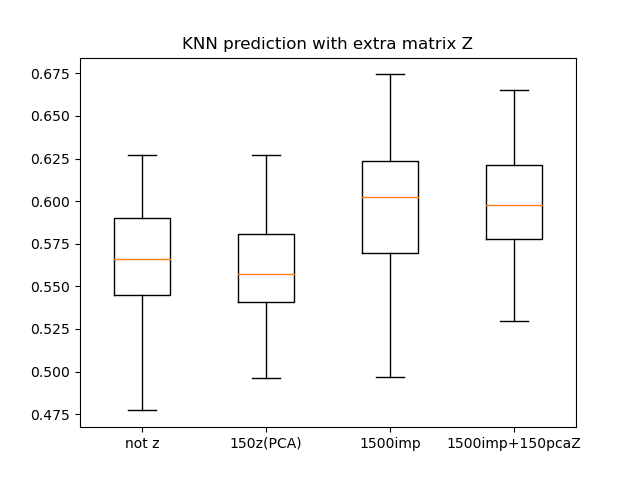
* 利用原始的X和Y进行拟合
* 利用X、Z和Y进行拟合
* 利用X、kernelPCA降维后的Z（因为矩阵Z过于稀疏）和Y进行拟合
* 利用重要的X（变量选择后的X）和Y进行拟合
* 利用重要的X、kernelPCA降维后的Z和Y进行拟合
* 利用重要的X、变量选择后的Z和Y进行拟合
* 利用重要的X去预测Y得到Y\_hat1，利用Z去预测残差Y-Y\_hat1得到Y\_hat2；最终预测结果为Y\_hat1+Y\_hat2（思路来自于文献Performance prediction of crosses in plant breeding through genotype by environment interactions），具体模型结构图如下：

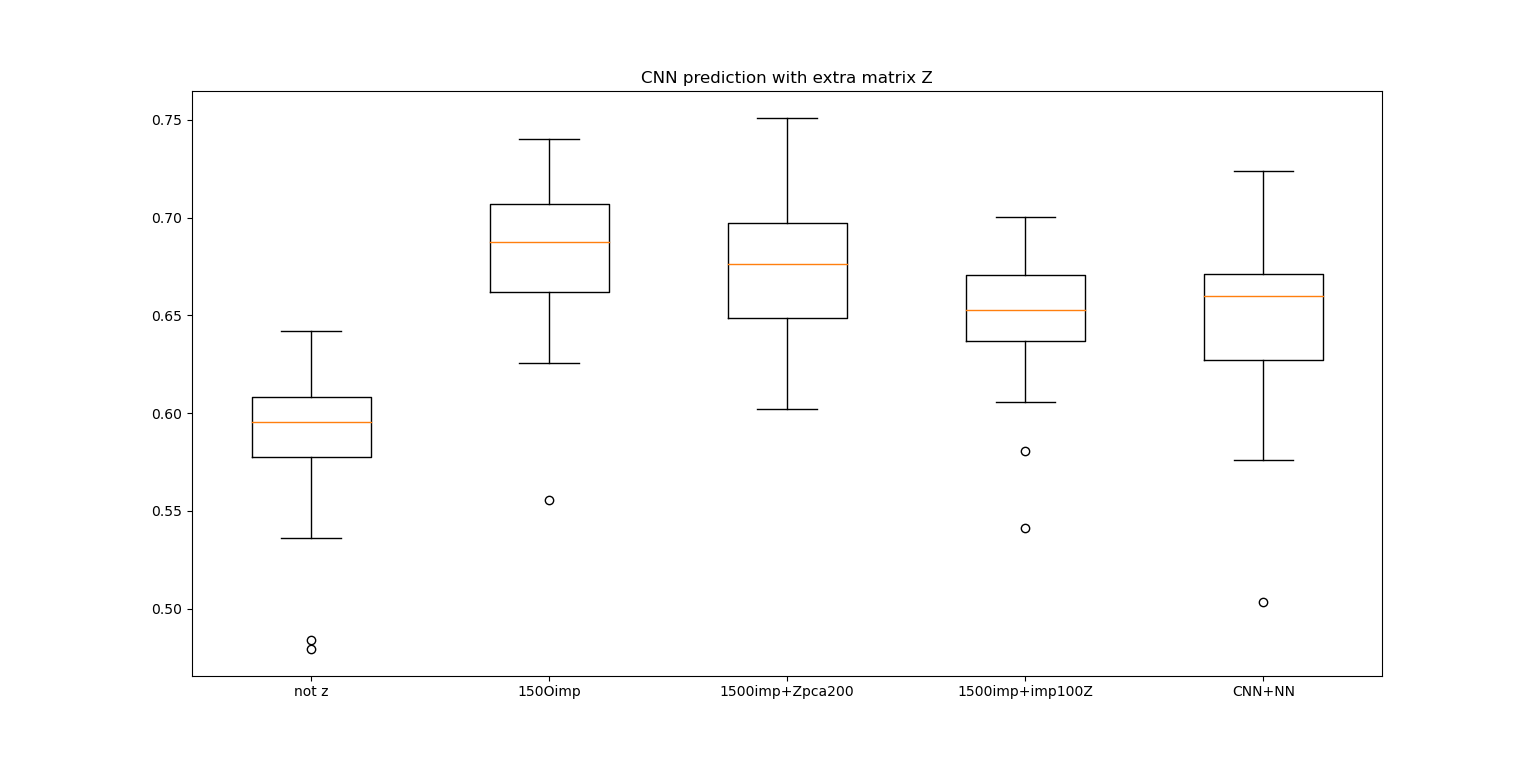


根据第二部分的分析结果，选择SVM、KNN、MLP和CNN进行试验探究。CNN由于运行时间过于漫长，每种情况只做了20次试验，其他算法做了50次独立重复试验。最终试验结果如下图所示：



上图中，“not Z”表示放入全部X没有放入Z是对比；“all Z”表示直接将X和Z拼接在一起放入模型训练；”150z(PCA)”表示利用KernelPCA对Z进行降维后再将其与X一起放入模型训练；“1500imp+1500impZ”表示选择出最重要的1500个X和最重要的1500个Z放入模型训练；“SVM+SVM”表示组合训练，第一个模型预测Y得到Y\_hat，第二个模型预测残差Y-Y\_hat。





由图可得如下结论：

* Z矩阵只能对普通机器学习算法（非深度学习算法）有一定的提升，对深度学习算法影响不大。因为深度学习本身就是一个强大的非线性拟合器，能隐含地捕捉到很多变量之间的交互效应。
* 相比于加入Z矩阵带来的信息，选择有效变量（进行变量选择）后再拟合模型对模型拟合精度的影响会更显著。
* Z矩阵的亲缘信息可能和SNP中的基因型信息有所重叠，导致对算法预测性能的提升不显著；根据先验，生物表型由基因和环境共同决定，加入环境信息可能会对模型预测性能有更显著的提升。