LAB01：Regression

小组编号：

小组成员 1：孙士晨

小组成员 2：李兵磊

小组成员 3：董祥虎

小组成员 4：聂　尧

姓名	学号	分工	占比
孙士晨	1853427	数据清洗、手写线性回归	25%
李兵磊	1852024	算法对比、交叉验证	25%
董祥虎	1850718	数据清洗、网格搜索	25%
聂　尧	1851909	特征工程、算法对比	25%
总　计			100%

目录

1.1. 数据准备

1.1.1. 数据集的获取

本次实验的数据集为 UCI 数据库的“Concrete Compressive Strength Data Set”，我们直接使用老师上传的数据文件 Concrete_Data.xls。

1.1.2. 数据预处理

为方便后续的代码编写和阅读，我们首先简化数据集的每个指标名，即去掉“成分”编号和指标单位。

	Cement	BlastFurnaceSlag	FlyAsh	Water	Superplasticizer	CoarseAggregate	FineAggregate	Age	CC_Strength
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.986111
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.887366
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.269535
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.052780
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.296075

1.2. 数据清洗

1.2.1. 缺失值补全

首先，我们查看数据集的缺失值情况，如有缺失值，需要补全。

```
Cement              0
BlastFurnaceSlag    0
FlyAsh              0
Water               0
Superplasticizer    0
CoarseAggregate     0
FineAggregate       0
Age                 0
CC_Strength         0
dtype: int64
```

可以看到数据集没有任何缺失值。

然后，我们将数据集分为 2 个集合——训练集和测试集。训练集和测试集又分别按照自变量和因变量分开。

```
X_train, X_test, y_train, y_test = train_test_split
                                (X, y, test_size=0.2, random_state=2021)
```

1.2.2. 标准归一化

之后，我们对训练集和测试集的因变量均进行了标准归一化，即

$$x_{std} = \frac{x - \mu}{\sigma}$$

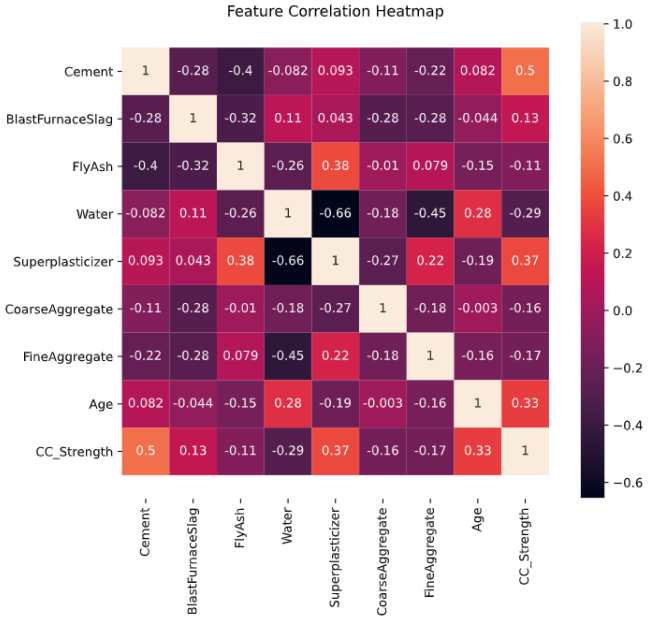
标准归一化有 2 个意义：

- 如果某个特征的方差比其他特征大几个数量级，那么它就会在训练过程中占据主导位置，导致其他特征在对训练结果的影像中占比很小，从而降低测试精度；

- 对数据进行标准归一化，可以加快梯度下降求解的速度。

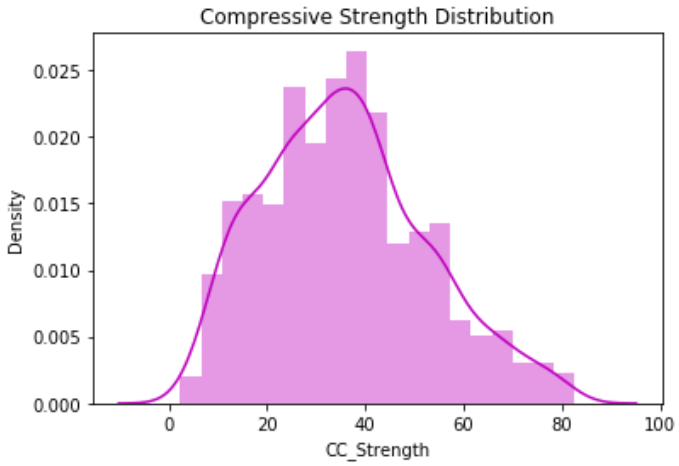
1.2.3. 特征工程

首先计算各个特征之间的相关系数，观察两两特征之间的关联程度。下图是两两特征的相关系数图：



可以看出，Cement、Superplasticizer 和 Age 对 CC_Strength 的影响最大。

然后棍叉因变量 CC_Strength 的分布，大致服从正态分布：



1.3. 模型搭建

我们采用了 9 种机器学习领域的回归模型，包括普通线性回归、Lasso 回归、Ridge 回归、支持向量机回归、K 最近邻回归、决策树回归、随机森林回归、Adaboost 回归和梯度提升树回归。下面简要介绍这些模型的数学形式、原理、优缺点及搭建过程。

其中， Lasso 回归、Ridge 回归均为普通线性回归的

1.3.1. 线性回归

1.3.1.1. 原理与数学形式

线性回归是数学形式为

$$Y = X\beta + \varepsilon$$

其中 Y 为响应变量，X 为协变量，β为回归系数，ε为随机误差项。线性规划的目标是寻求一个合适的回归系数β，使得误差最小。

当 X 为列满秩矩阵时，回归系数 β 可由普通的最小二乘估计方法求得

$$\hat{\beta}_{OLS} = \arg\min_{\beta \in R^d} |Y - X\beta|^2 = (X^T X)^{-1} X^T Y$$

1.3.1.2. 特点

- 优点：有闭合形式的解，可以直接计算，无需迭代训练。
- 缺点：对数据要求较高，若 X 不是列满秩矩阵，则不能使用最小二乘法求解回归系数。

1.3.1.3. 搭建过程

调用 sklearn 库函数即可建立新型回归模型。

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

1.3.2. LASSO 回归

1.3.2.1. 原理与数学形式

当矩阵 X 不满足列满秩时，将不能采用普通最小二乘法来求解回归系数，此时引入惩罚方法。Lasso 方法是在普通线性模型中增加 L 惩罚项，对于普通线性模型，Lasso 估计为

$$\hat{\beta}_{Lasso} = \arg\min_{\beta \in R^d} (|Y - X\beta|^2 + \lambda \sum_{j=1}^{d} |\beta_j|)$$

1.3.2.2. 特点

可以将一部分系数压缩到 0，从而降低 X 的维度，达到减小模型复杂度的目的。

1.3.2.3. 搭建过程

调用 sklearn 库函数即可建立新型回归模型。

```
from sklearn.linear_model import Lasso
lasso = Lasso()
```

1.3.3. RIDGE 回归

1.3.3.1. 原理与数学形式

当协变量$X^{(j)}$间相互独立时，采用普通最小二乘法估计参数 β 具有很好的性质。但是当协变量维度增加时，$X^{(j)}$间难免会存在相关关系，此时设计矩阵 X 将不再满足列满秩，我们称这样的设计矩阵为病态的。当 X 呈病态时，X^TX接近奇异，此时$\hat{\beta}_{OLS}$虽然具有最小方差，但是$\hat{\beta}_{OLS}$值很大，导致精度比较差，表现相当不稳定。针对设计矩阵为病态的情况，众多学者提出了不同的改进方法，用得较为广泛的是 Ridge 方法，也称为岭回归。对于线性模型，Ridge 回归的数学形式为

$$\hat{\beta}_{Ridge} = arg\min_{\beta \in R^d}\left(|Y - X\beta|^2 + \lambda\sum_{j=1}^{d}|\beta_j|^2\right) = \frac{1}{1+\gamma}\widehat{\beta_{oLS}}$$

1.3.3.2. 特点

Ridge 估计是对普通最小二乘估计以同一比例进行压缩，但是不会将原本非零的系数压缩至 0。

1.3.3.3. 搭建过程

调用 sklearn 库函数即可建立新型回归模型。

```
from sklearn.linear_model import Ridge
ridge = Ridge()
```

1.3.4. 支持向量机

1.3.4.1. 原理与数学形式

SVM 的学习策略是间隔最大化，可形式化为一个求解凸二次规划的问题，也等价于正则化的合页损失函数的最小化问题。SVM 的学习算法就是求解凸二次规划的最优化算法。SVM 回归的目标函数为

$$\min_{w,b} C\sum_{i=1}^{N} E_\epsilon(\hat{y}_i - y_i) + \frac{|w|^2}{2}$$

1.3.4.2. 特点

- 优点：VM 学习问题可以表示为凸优化问题，因此可以利用已知的有效算法发现目标函数的全局最小值。
- 缺点：SVM 算法对大规模训练样本难以实施。

1.3.4.3. 搭建过程

```
from sklearn import svm
SVM = svm.SVR()
```

1.3.5. K 最近邻算法

1.3.5.1. 原理与数学形式

K 最近邻算法又称为 KNN 算法。所谓的 K 最近邻，就是指最接近的 K 个邻居（数据），即每个样本都可以由它的 K 个邻居来表达。

kNN 算法的核心思想是，在一个含未知样本的空间，可以根据离这个样本最邻近的 k 个样本的数据类型来确定样本的数据类型。

1.3.5.2. 特点

- 优点：模型训练时间快;
- 缺点：对不相关的功能和数据规模敏感。

1.3.5.3. 搭建过程

```
from sklearn import neighbors
KNN = neighbors.KNeighborsRegressor()
```

1.3.6. 决策树回归

1.3.6.1. 原理与数学形式

决策树的输入训练数据是一组带有类别标记的样本，构造结果是一棵多叉树。树的分支节点一般表示为一个逻辑判断。某一节点上选择特征的标准是在该节点上选取能对该节点处的训练数据进行最优划分的属性。划分的标准是信息增益（Information Gain），即划分前后数据集的熵的差异。取能带来最大信息增益的那个特征作为当前划分标准。

信息熵

$$\mathrm{Info}(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

信息增益

$$\mathrm{Info}_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \mathrm{info}(D_j)$$

1.3.6.2. 特点

- 优点：效率高，在相对短的时间内能够对大型数据源做出可行且效果良好的结果。
- 缺点：容易过拟合、对连续性字段比较难预测。

1.3.6.3 搭建过程

调用 sklearn 库函数即可建立新型回归模型。

```
from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor()
```

1.3.7. 随机森林回归

1.3.7.1. 原理与数学形式

随机森林属于 Bagging 类算法，而 Bagging 方法又属于集成学习一种方法，集成学习的大致思路是训练多个弱模型打包起来组成一个强模型，强模型的性能要比单个弱模型好很多，其中的弱模型可以是决策树、SVM 等模型，在随机森林中，弱模型选用决策树。

在训练阶段，随机森林采样从输入训练数据集中采集多个不同的子训练数据集来依次训练多个不同决策树；在预测阶段，随机森林将内部多个决策树的预测结果取平均得到最终的结果。

1.3.7.2. 特点

- 在当前所有算法中，具有极好的准确率；
- 能够有效地运行在大数据集上；
- 能够处理具有高维特征的输入样本，而且不需要降维；
- 能够评估各个特征在分类问题上的重要性。

1.3.7.3 搭建过程

调用 sklearn 库函数即可建立新型回归模型。

```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(n_estimators=100)
```

1.3.8. ADABOOST 算法

1.3.8.1 原理与数学形式

Adaboost 是一种迭代算法，其核心思想是针对同一个训练集训练不同的分类器(弱分类器)，然后把这些弱分类器集合起来，构成一个更强的最终分类器（强分类器）。

1.3.8.2. 特点

- 优点：
  - 可以将不同的分类算法作为弱分类器；
  - 相对于随机森林算法，Adaboost 充分考虑的每个分类器的权重。
- 缺点：
  - 数据不平衡导致分类精度下降。
  - 训练比较耗时，每次重新选择当前分类器最优切分点。

1.3.8.3. 搭建过程

```
from sklearn.ensemble import AdaBoostRegressor
Ada = AdaBoostRegressor(n_estimators=100)
```

1.3.9. 梯度提升树回归

1.3.9.1. 原理与数学形式

BDT 也是集成学习 Boosting 家族的成员，但是却和传统的 Adaboost 有很大的不同。Adaboost 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。GBDT 也是迭代，使用了前向分布算法，但是弱学习器限定了只能使用 CART 回归树模型，同时迭代思路和 Adaboost 也有所不同。

在 GBDT 的迭代中，假设前一轮迭代得到的强学习器是$f_{t-1}(x)$，损失函数是$L(y, f_{t-1}(x))$， 则本轮迭代的目标是找到一个 CART 回归树模型的弱学习器$h_t(x)$，让本轮的损失函数$L(y, f_t(x)) = L(y, f_{t-1}(x) + h_t(x))$最小。

1.3.9.2. 特点

- 优点：
  - 天然就可处理不同类型的数据；
  - 对空间外的异常点处理非常健壮。

■ 缺点： 扩展性不好，难以并行化。

1.3.9.3. 搭建过程

```
from sklearn.ensemble import GradientBoostingRegressor
GBRT = GradientBoostingRegressor(n_estimators=100)
```

1.4. 模型训练测试

9 种回归模型的训练过程：

```
lr.fit(X_train, y_train)
lasso.fit(X_train, y_train)
ridge.fit(X_train, y_train)
SVM.fit(X_train, y_train)
KNN.fit(X_train, y_train)
Ada.fit(X_train, y_train)
dtr.fit(X_train, y_train)
rfr.fit(X_train, y_train)
GBRT.fit(X_train, y_train)
```

9 种回归模型的测试过程：

```
y_pred_lr = lr.predict(X_test)
y_pred_lasso = lasso.predict(X_test)
y_pred_ridge = ridge.predict(X_test)
y_pred_SVM = SVM.predict(X_test)
y_pred_KNN = KNN.predict(X_test)
y_pred_Ada = Ada.predict(X_test)
y_pred_dtr = dtr.predict(X_test)
y_pred_rfr = rfr.predict(X_test)
y_pred_GBRT = GBRT.predict(X_test)
```

1.5. 结果可视化

我们主要用 4 个指标来衡量不同及其学习方法的优劣，即 RMSE、MSE、MAE 和 R2 score。下面展示我们采用的 4 个指标的数学形式，其中，y_{test}（y）和$\hat{y}_{test}$（$\hat{y}$）分别表示真实测试数据和预测训练数据。

均方误差 MSE（Mean Squared Error）：

$$\frac{1}{m}\sum_{i=1}^{m}\left(y_{test}^{(i)}-\hat{y}_{test}^{(i)}\right)^2=MSE$$

均方根误差 RMSE（Root Mean Squared Error）：

$$\sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(y_{test}^{(i)}-\hat{y}_{test}^{(i)}\right)^2}=\sqrt{MSE_{test}}=RMSE_{test}$$

均绝对误差 MAE（Mean Absolute Error）：

$$\frac{1}{m}\sum_{i=1}^{m}\left|y_{test}^{(i)}-\hat{y}_{test}^{(i)}\right|=MAE$$

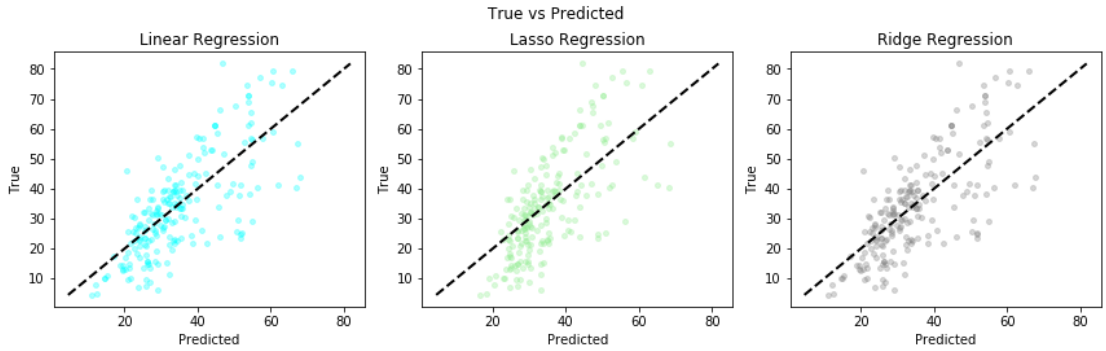
R 方值（R2_score）：

$$R^2=1-\frac{\sum_i\left(\hat{y}^{(i)}-y^{(i)}\right)^2}{\sum_i\left(\bar{y}-y^{(i)}\right)^2}$$

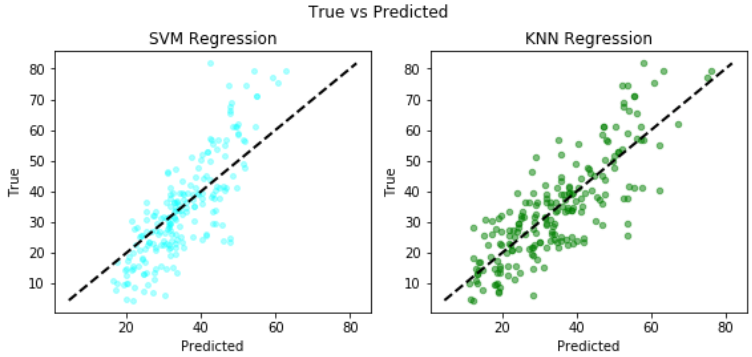
9 种模型的 4 个指标的实验结果如下表：

模型	RMSE	MSE	MAE	R2
线性回归	11.17	124.81	8.79	0.548376
Lasso 回归	11.97	143.22	9.56	0.481768
Ridge 回归	11.17	124.85	8.80	0.548220
SVM 回归	10.27	105.39	8.05	0.618657
KNN 回归	9.55	91.14	7.5	0.670206
决策树回归	7.25	52.49	4.72	0.801340
随机森林回归	5.22	27.26	3.66	0.896802
Adaboost 回归	7.91	62.50	6.48	0.773860
梯度提升树回归	5.35	28.62	3.88	0.896433

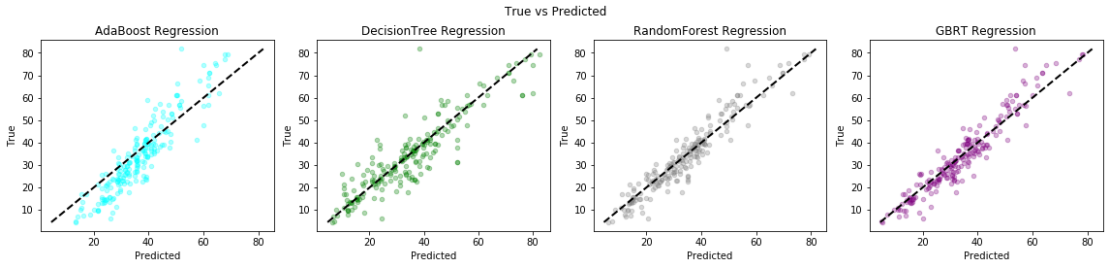
下图是 3 个线性模型的以预测值为横轴、真实值为纵轴的散点图。如果散点离对角线越近，则说明模型效果越好。可以看出，普通线性回归和 Ridge 回归的效果比 Lasso 回归要好，这是因为 Lasso 算法简化了模型，使得很多系数为 0，降低了准确率。



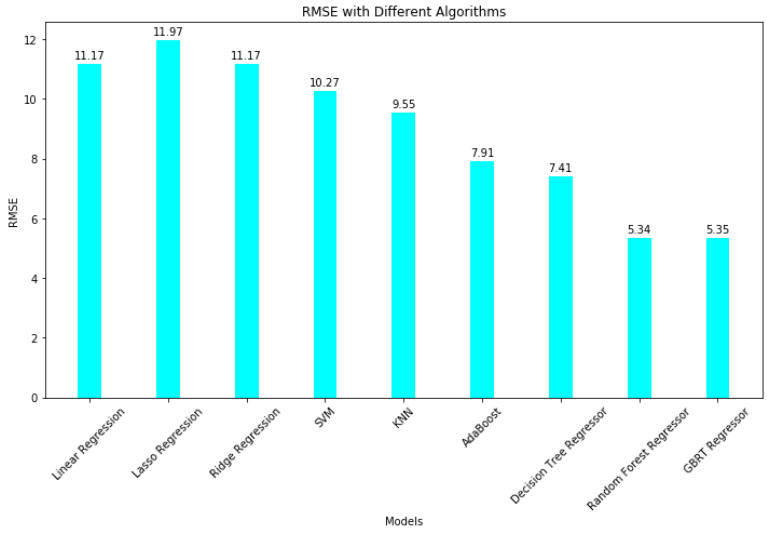
下图是 SVM 回归和 KNN 回归的散点图。



下图是决策树、随机森林、Adaboost 和梯度上升树的散点图。

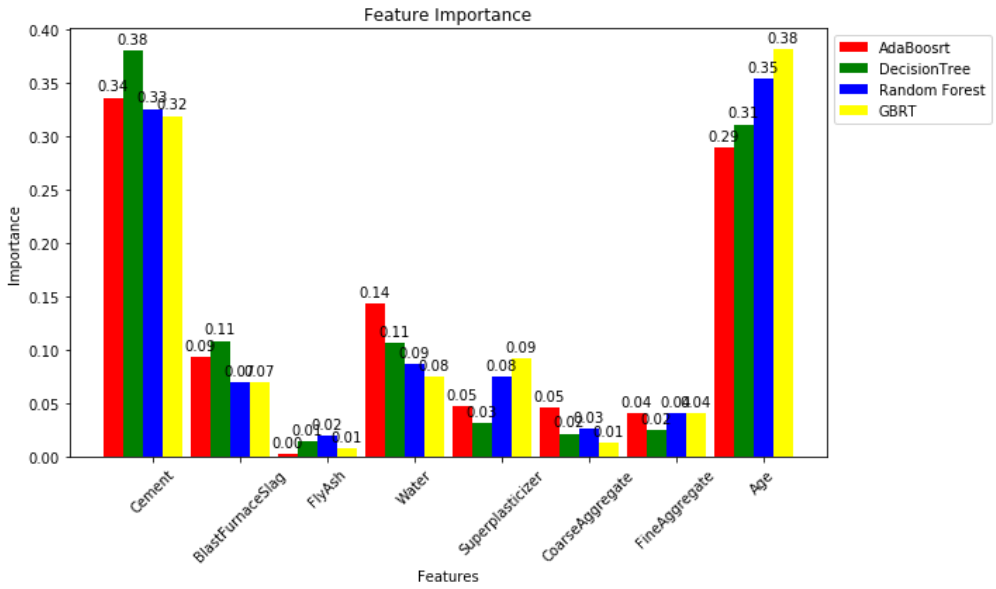


对比所有模型的 RMSE:



可以看出，随机森林的 RMSE 最小，最适合本问题。

树类回顾算法可以生成 Feature Importance，来表示不同特征对因变量的影响程度。下图是 Feature Importance：



可以看出，Cement 和 Age 被视为最重要的特征。在预测混凝土强度时，Flyash、Coarse 和 Fine 是最不重要的因素。下面去掉 FlyAsh 和 Fine2 个特征，选用随机森林回归模型，观察实验结果：

模型	RMSE	MSE	MAE	R2
全特征	5.22	27.26	3.66	0.896802
6 特征	5.19	26.90	3.59	0.900874

可以看到在去掉 2 个特征后，R2 值升高，而 RMSE、MSE 和 MAE 小于全特征，说明这两个特征属于噪声，去掉后模型精度更高。

接下来用普通线性回归处理少特征数据。根据前面的部分，我们可以得到，Fly Ash，Coarse Aggregate，Fine Aggregate 三个特征和 Concrete compressive strength 的相关性较差。

首先，使用 sklearn 库函数进行特征选取，选择 6 个最好的特征，

```
[781] ▷ ▸≡ M↓
    from sklearn.feature_selection import SelectKBest,f_regression,chi2,f_classif,
    mutual_info_regression
    # f_regression计算相关性
    x_new = SelectKBest(f_classif,k=6).fit(X_train,y_train).get_support(indices=True)
    print(x_new)
  [0 1 2 3 4 7]
```

进行测试，得到如下的结果

模型	RMSE	MSE	MAE	R2
全特征	9.69	93.85	7.71	0.658452
6 特征	9.67	93.60	7.72	0.659353

可以看到，减少了两个变量之后，模型的精确度几乎没有影响。

分析原理：考虑到线性规划的核心思想$y = \theta X$，当一个变量对 y 的相关性较弱的时候，这个变量在这个式子中的作用也较小。当相关性较弱，即 x 的一个分量的改变对 y 的影响较小，这种时候删去这个变量，能够减少计算量，还不影响精确度。

1.6. 模型优化

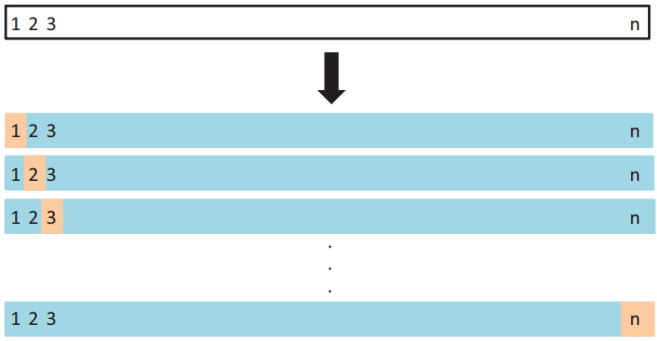
分析实验模型的优缺点并进行模型优化改进

1.6.1. 交叉验证

1.6.1.1. 原理

我们使用的是 k 折交叉验证方法。如图，假设有 n 个数据组成的数据集，我们将其划分为 k 个等大的数据子集，每次取出一个数据子集作为测试集，而其他 n-1 个数据子集都作为训练集用于训练模型和调参。结果就是我们最终训练了 k 个模型，每次都能得到一个 MSE。而计算最终 test MSE 则就是将这 k 个 MSE 取平均，即

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$



1.6.1.2. 特点

交叉验证充分利用了数据集的每一个数据，不会过拟合训练集而不适用于测试集，训练出的模型具有较好的泛化能力。

1.6.1.3. 结果

Order	RMSE	MSE	MAE	R2
1 of 5 folds	9.78	95.64	7.87	0.636898
2 of 5 folds	10.46	109.40	8.26	0.592934
3 of 5 folds	11.20	125.40	8.87	0.610407
4 of 5 folds	10.20	104.09	8.12	0.634528
5 of 5 folds	9.92	98.31	7.96	0.651915

上图是 5 折交叉验证的 4 个评价指标。对 R2-score 取平均，对比得

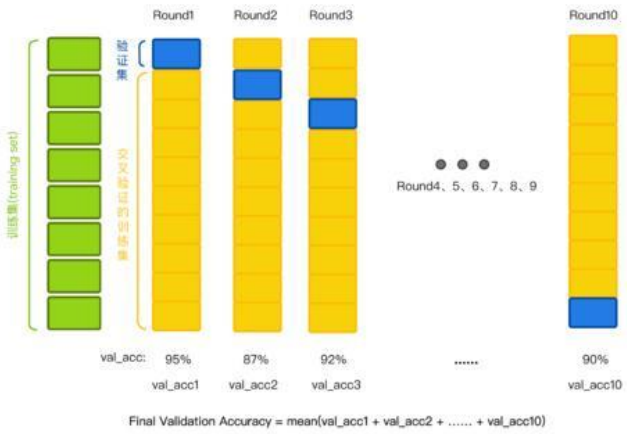
- 不采用交叉验证的普通线性回归模型的 R2-score：0.548376；
- 采用 5 折交叉验证的普通线性回归模型的平均 R2-score：0.625337。

决定系数（R-square）提高了 14.03 %，说明交叉验证模型优化效果显著。

1.6.2. 网格搜索

1.6.2.1. 原理

网格搜索（GridSearchCV）的名字其实可以拆分为两部分，GridSearch 和 CV，即网格搜索和 1.6.1 使用的交叉验证。网格搜索，搜索的是参数，即在指定的参数范围内，按步长依次调整参数，利用调整的参数训练学习器，从所有的参数中找到在验证集上精度最高的参数，这其实是一个训练和比较的过程。GridSearchCV 可以保证在指定的参数范围内找到精度最高的参数，但是这也是网格搜索的缺陷所在，他要求遍历所有可能参数的组合，在面对大数据集和多参数的情况下，非常耗时。



1.6.1.2. 特点

网格搜索适用于三四个（或者更少）的超参数（当超参数的数量增长时，网格搜索的计算复杂度会呈现指数增长，这时候则使用随机搜索），用户列出一个较小的超参数值域，这些超参数

至于的笛卡尔积（排列组合）为一组组超参数。网格搜索算法使用每组超参数训练模型并挑选验证集误差最小的超参数组合。

1.6.1.3. 结果

Model	RMSE	MSE	MAE	R2
KNeighborsRegressor	9.55	91.14	7.52	0.67

Model	RMSE	MSE	MAE	R2
GridSearch	8.55	73.08	6.38	0.74

对于普遍使用的线性回归函数 LinearRegression()，由于其线性回归的特殊约束性，我们一般对其使用特征选择-RFECV 优化。而为了验证网格搜索的优化属性，我们选择了 KNN 回归（KneighborsRegressor）进行验证。它的原理是：一个样本在特征空间中 k 个最相似（最邻近）的样本大多数属于同一个类别。使用它可以解决分类和回归问题。

上图 R2 是评价指标。对比得：

- 不采用网格搜索的普通 KNN 模型的 R2 为：0.67；
- 采用网格搜索的 KNN 模型的 R2 为：0.74。

决定系数（R-square）提高了 10.4 %，说明使用网格搜索对包括 KNN 在内的一些回归算法效果显著。

EXTRA CREDIT：手写实现线性回归算法模型

我们手写实现了线性回归算法模型，用同样的数据集训练了手写和 sklearn 库两种模型实现，并进行了对比。我们手写实现的线性回归算法模型主要有一下特点：

- 只调用了 numpy 库，其余均通过 python 基本语法完成；
- 实现了数据集的标准化；
- 选用梯度下降法求解系数；
- 同 sklearn 库的使用方法类似，可以自定义学习率和迭代次数，调用 my_fit(x,y)即可进行训练。

EC.1. 数学原理

线性回归的数学形式为

$$y = \theta^T X$$

θ为线性回归的系数。

为了描述线性回归的误差，我们定义了如下均方差指标，即

$$MSE = \frac{1}{m}\sum_{i=1}^{m}\left(\theta^T X^{(i)} - y^{(i)}\right)^2$$

其中，$X^{(i)}, y^{(i)}$代表第 i 个样本的自变量和因变量。

线性回归的目标是通过某种优化方法找到一个回归系数θ，使 MSE 达到最小。

我们采用的优化方法是梯度下降法。对于 θ 的第 i 个分量θ_i，对 MSE 求偏导有

$$\frac{\partial MSE(\theta)}{\partial \theta_i} = \frac{2}{m} * \sum_{i=1}^{m}\left(\theta^T X^{(i)} - y^{(i)}\right) * x^{(i)}$$

梯度为

$$\frac{\partial MSE(\theta)}{\partial \theta} = \begin{pmatrix} \frac{\partial MSE(\theta)}{\partial \theta_1} \\ \frac{\partial MSE(\theta)}{\partial \theta_2} \\ \vdots \\ \frac{\partial MSE(\theta)}{\partial \theta_m} \end{pmatrix} = \frac{2}{m} X^T \cdot (X \cdot \theta - y)$$

EC.2. 实现原理

我们通过分析主要函数说明实现原理。

```
def normalized(X)
```

- 功能：归一化函数。
- 输入：n 个样本。
- 输出：标准化后的 X、用于转化测试集的平均值 mu 和方差 sigma。
- 内容：通过$\frac{\mathrm{X-mu}}{\mathrm{sigma}}$来完成标准化。

```
def GradientDescent(X,y,theta,alpha,iters)
```

- 功能：梯度下降函数。
- 输入：标准化且添加常数项后的 X、对应的 y、回归系数 theta 的初始值、每次下降的步长 alpha、最大迭代次数 iters。
- 输出：回归系数 theta 的近似最优解和历史的误差变化。

- 内容：每次迭代求梯度，theta 向着下降最快的方向（即梯度方向）移动，直到前后两次误差相差不大或者达到最大迭代次数。

```
def my_fit(X,y,alpha=0.01,iters=500)
```

- 功能：拟合函数。
- 输入：标准化后的 X、对应的 y、迭代步长 alpha、最大迭代次数 iters。
- 输出：回归系数 theta 的近似最优解和历史的误差变化。
- 内容：输入 X、y，调用梯度下降函数，求解 theta。

```
def predict(X,mu, sigma, theta)
```

- 功能：预测函数。
- 输入：原始样本输入 X、训练集标准化所需的 mu 和 sigma、训练得到的回归系数 theta。
- 输出：预测值。
- 内容：将 X 标准化后，加入常数项，根据$y = \theta^T X$计算预测值。

EC.3. 实验结果

我们定义误差函数为

$$m = \frac{1}{n}\sum_{1}^{n}(y_{pred} - y_{true})^2$$

然后分别调用 sklearn 的线性回归库函数和自己手写的线性回归函数，输出二者的预测误差和运行时间，进行对比。对比结果如下图：

```
sklearn-mse [124.80940516]
use time   0:00:00.252204
my-mse [130.63234323]
use time   0:00:00.016956
```

可以看到，手写方法和 sklearn 库函数误差相差不大，而运行时间远小于 sklearn 库函数，约小了一个数量级。

EC.4. 优化方向

手写的线性回归还有一些待优化的方向，具体如下：

- 优化采用了梯度下降的方法，本身是一个局部最优解，不一定达到全局最优；
- 梯度下降法是一阶收敛的优化方式，如果采用牛顿法等二阶优化方式，求解速度更快；

- 对数据的处理采用了标准化的方法，可以尝试归一化是否有更好的结果。

1.7. 优化方向

本次实验中，采用的回归方式为线性回归，没有考虑到高次函数，或者指数对数的情况。因为没有混凝土的相关物理知识，不知道哪里变为何种非线性的情况更好，进一步优化我们可以考虑进一步的特征工程，将自变量进行变换，变成高次或指对数，再使用线性回归。